Final Report

Using indoor location technology to create an interactive experience for users through a mobile application

---

Author: Stuart Jones

Supervisor: Chris Jones
Moderator: Omer Rana

CM3202 - One Semester Individual Project - 40 Credits
May 2015

Abstract

This project involved evaluating the operating characteristics of iBeacons. These findings were used to develop an indoor positioning system that allowed users to interact with the system using their physical location. The system included a mobile application that interacted with the iBeacons to determine how close the user is to specific points in the room and delivered content to users. A system management web interface was also developed allowing user behaviour to be tracked and other content to be delivered.

I worked in collaboration with Hoffi whom I worked part time for to deliver this system for Yellobrick. Yellobrick customers use the mobile application and Yellobrick employees use the system management interface to track user behaviour.

Acknowledgements

I would like to express special gratitude to my final year project supervisor, Dr. Christopher Jones, whose advice and guidance helped me throughout my project especially in writing this report.

Furthermore I would also like to acknowledge with much appreciation the crucial role of Julian Sykes and Andrew Thomas of Hoffi, who played a major role in this project as without them this project would never have taken place. I would also like to thank them for giving me permission to use their iBeacons and servers.

A special thanks goes to Alison John on behalf of Yellobrick who were a pleasure to work with for the duration of this project.

Last but not least, many thanks go to my friends, family and girlfriend who have supported and encouraged me throughout this project.

Contents

Figures

Tables

Code listings

1 Introduction

Interacting with users based on their indoor location provides many possibilities. Augmented reality gaming where the user's physical location information is used to combine the virtual and real world is just one of these possibilities. Others include locating and tracking both people and objects indoors which has endless uses including navigation, advertising and finding objects.

Indoor Positioning Systems (IPS) have always been a challenge to create with there being many drawbacks including ever changing environments, but with endless opportunities many companies including Apple, Google and Microsoft are investing heavily in this industry (Costa, 2013).

Most people in the United Kingdom now own smartphones (Brenchley, 2014) this has made interacting with IPS more accessible than ever. These technologies can be combined to deliver content to users in real-time based on their physical indoor location and allow organisations to track users physical location behaviour.

I will be working in collaboration with Hoffi (Hoffi, 2012) whom I work part time for. Hoffi are a branding company based in Cardiff Bay who develop a brand strategy, identity and communications for businesses and organisations through a variety of methods including web and mobile development which is the area I work in.

Hoffi were contacted by a company called Yellobrick (Yellobrick, 2014). They are a creative marketing agency that build engaging and participatory experiences for brands and organisations. They asked Hoffi to develop a user-friendly indoor system which would allow users to interact with it using their physical location. A mobile application will be used to tell the user how close they are to areas of interest and a system management interface will be needed so the system management team can track users and set up content for users. The project was set in December 2014 with a deadline of March 2015. This timeframe linked in well with my final year project. Once the project was finished the system was presented to the client in March when detailed feedback was received.

In the background section I will mention in detail what positioning systems and indoor positioning systems are. I will also review current indoor technologies where I will evaluate the different technologies and compare them to iBeacons.

To develop this prototype I will use the still relatively new Bluetooth IPS technology iBeacon developed by Apple in 2013. The iBeacons work by sending out Bluetooth low energy (BLE) signals to communicate with smartphones and other devices that support BLE (Ratcliff, 2014). The BLE signal strength is used to calculate how far away a device is from a beacon.

The project will be split into a two-stage development process. The first stage will involve developing and evaluating the operating characteristics of iBeacons to find out the most accurate, efficient and suitable way to position a user indoors using iBeacon technology. This will be the experimental stage allowing me to undsertand how iBeacons work. I will then use these findings to help with the second stage of the project developing the Yellobrick system.

The Yellobrick system will be split into two sub-systems. A mobile application will be used by the Yellobrick customers and a system management interface will be used by Yellobrick employees. The mobile application will interact with the iBeacons to calculate the users position and then send this information to the server when location conditions are met. This application will also tell the user how far they are from each of the important locations and notify them if they enter a specific area. The system management interface will allow Yellobrick employees to view user behaviour and set up content on computer monitors. This content will vary depending on user behaviour. Some content will trigger when users enter specific locations and other content will only trigger when the user visits the locations in a specific order. Another part of the interface will involve a user walking close to a projected digital image and then this digital image will start moving. If another user gets close to the painting it will stop moving. The purpose of this is to confuse the user in a dark horror scene type setting. A database stored on the Hoffi server will be used so the two systems can interact with the same data.

The main scope of this system is to deliver a working system that allows users to interact with the system via a mobile application. The system will only be available locally so security won't be a priority. As this is just a prototype demonstrating the potential of indoor positioning systems using iBeacons it is only essential it works on iOS, but in the future it would be nice if it worked across the two major platforms iOS and Android.

The main aims and objectives of this project:

- Evaluate the IPS iBeacon technology and compare it with other IPS technologies.
- Investigate iBeacons to find the best way to accurately determine the position of a user in real time, deliver content to users at specific locations and track user "dwell times" anonymously at specific locations.

- Deliver a working prototype for Yellobrick on behalf of Hoffi that allows the user to interact with it using their physical location and Yellobrick employees to track user behaviour. This will showcase the different

2 Background

2.1 Positioning Systems

A positioning system is a mechanism used for determining the location of an object in space. There are many technologies used for this ranging from worldwide coverage with metre accuracy to workspace coverage with sub millimetre accuracy. These systems are split into four main categories global, regional, sitewide and workspace systems.

Global systems allow specialised radio receivers to determine the 3D space position and time of an object usually within an accuracy of 20 metres. Current deployed systems such as Global Positioning System (GPS) use microwave signals that are only reliable outdoors.

Regional systems are networks of land-based positioning transmitters that are used to determine the 2D position of an object on the surface of the Earth. These are generally less accurate than global systems as their signals are not entirely restricted to line of sight propagation and only have regional coverage. An example system would be Long-range navigation (LORAN) (Robert Lilley, 2008) which provides ranges up to 1500 miles with an accuracy of 10 miles.

Sitewide systems are known as indoor positioning systems that can be used within individual rooms, buildings or construction sites. They usually offer centimetre accuracy and are often used in places where global and regional systems aren't suitable. An example would be Google's indoor map system (Google, 2014).

Workspace systems are designed to cover a restricted space, usually a few cubic metres. This type of system can offer very high accuracy in the millimetre-range or better. An example system would be the Wii Remote (Amazon, 2007).

IPS systems offer a lot of potential use cases, but are not as advanced as global and workspace systems.

2.2 Indoor Positioning Systems (IPS)

An IPS is a solution that allows objects or people to be located in an indoor environment (Kevin Curran, 2011).

IPS provide location services in places where satellite signals are attenuated, especially inside buildings. The complexity of buildings and other indoor environments affect the transmission of satellite signals. This means the GPS can detect that an object is within a building, but it cannot determine exactly where inside. Figure 1 shows a visual comparison between GPS and IPS.

Figure 1: GPS and IPS comparisons (Anthony, 2012)

IPS can be used for many different applications. These applications are split into two main categories navigation and tracking. IPS navigation can be used in lots of use cases. It can be used to help visually impaired people navigate buildings. Large indoor environments such as a supermarket IPS can provide a navigation service to users to help them find specific groceries. IPS tracking can be used in many more use cases including location based advertising in stores so when a customer spends a lot of time by a specific pair of shoes an advert could be sent to the user showing similar shoes. Other examples are: tracking children in a busy area such as a mall so when a child leaves your sight you can still find them using your application; allowing gamers to bridge their real world environment with the virtual world using their physical

location; at an art gallery where information about a painting could be displayed to the user if they were close to it. These examples show the huge potential of IPS and what they can offer us in society.

There are two types of location services, absolute location and relative location. Absolute location refers to geo location, for example your latitude and longitude. Relative location defines your position relative to real world objects, for example you are one metre from that door (Williams, 2014).

There are a lot of different technologies that can be used for IPS. These will be discussed in section 3. For this project I will be using Bluetooth LE technology which focuses on proximity and not absolute location therefore allowing us to find a mobile clients' relative location (Aerohive Networks, Inc., 2014).

2.3 Bluetooth Low Energy (BLE)

The main feature of BLE is its lower power consumption making it possible to power a small device using a tiny coin cell battery. Power consumption is kept low as it is asleep most of the time only activating when a connection is initiated. The technology works by sending BLE signals from a beacon to a receiver. A calculation is then performed by the receiver to calculate the distance between the two. More details about BLE can be found in section 3.8. Apple was the first smartphone company to support BLE on its iPhone 4S in 2011, who were followed by Microsoft, Android and Blackberry. Apple announced iBeacons in the summer of 2013 (Hern, 2014).

2.4 iBeacons

IBeacon is Apple's implementation for an indoor positioning system which provides a proximity location service using Received Signal Strength Indication (RSSI) calculation. The iBeacon signal contains an address which is used to identify the beacon. The address consists of 3 components which are explained in Table 1 (Baskerville, 2014).

Table 1: Bluetooth low energy address components

Component	Description	Usage	Example
ProximityUUID	A 128 bit string that represents a group of beacons.	Identify a company.	B9407F30-F5F8-466E-AFF9-25556B57FE6D
Major	A 16 bit unsigned integer value ranging from 0 up to 65536 used to identify beacons using the same ProximityUUID.	Identify a building or branch within a company.	62361
Minor	A 16 bit unsigned integer value ranging from 0 up to 65536 used to identify beacons using the same ProximityUUID and Major.	Identify a region or room within a building.	4291

The iBeacon signal also has three power characteristics which are described in Table 2 (Puchta, 2014a).

Table 2: iBeacon power characteristics

Power Characteristics	Description
Broadcasting power	The power the beacon broadcasts its signal at. Value range from -30 dBm up to +4dBm. The higher the broadcasting power the more stable the signal, but the shorter the battery life.
RSSI	This is the strength of the beacon's signal as seen on the receiving device. In general the higher the distance between the beacon and the receiver the lesser the strength of the received signal.
Measured power	This is a factory-calibrated, read only constant that indicates what the expected RSSI at a distance of 1 metre from the beacon should be.

These three values are used to estimate the distance between the beacon and the receiver. Due to external factors such as absorption, interference or diffraction this RSSI value tends to fluctuate which can result in unstable readings and therefore less accurate estimates. Overall

Bluetooth LE and iBeacon technology is fairly new, but it shows huge potential. It's very accurate, available on most modern smartphones, easy to set up, efficient and cheap. It has a few problems regarding interference, but in the right environment could be very powerful.

2.5 Estimote iBeacons

There are many different iBeacon vendors. Aislelabs conducted a survey on sixteen of the most popular vendors (Aislelabs, 2014). In this report they state Estimote is the most stylish iBeacon and speak of it highly. I chose to use Estimote iBeacons (Estimote, 2013) as they are one of the best-known beacon vendors. The company distributed over 10,000 in the first 6 months (Thompson, 2013) and has a very active community that provides great support. They are constantly updating and improving their hardware and APIs which is demonstrated by their recent release of the WatchKit API (Piotr Krawiec, 2015).

3 Indoor Positioning Technologies Review

Here I will review the current indoor positioning technologies. There are two main categories of IPS, non-radio technologies and radio technologies. Non-radio technologies such as magnetic positioning don't require existing wireless infrastructure. Any wireless radio technology can be used for locating an object or user such as Wi-Fi or Bluetooth. Wireless radio technology position accuracy can usually be increased at the expense of wireless infrastructure equipment and installations.

3.1 Magnetic Positioning

Magnetic positioning is based on the iron inside buildings that create local variations of the Earth's magnetic field. Compass chips inside smartphones can sense and record these variations to map indoor locations (Hexagon, 2014). The benefits of this technology are that it requires no additional hardware, has 1-2 metres accuracy, 90% precision and is available on most smart phones. The problems with this technology are that it requires a very precise floor plan of the entire room or building to set up which can take a long period of time (Owano, 2012) and it has very little documentation. Overall the magnetic positioning system IPS is very clever and has a lot of potential, but it needs much more in depth testing and is difficult to produce on a large scale due to the amount of time it takes to set up.

3.2 Global Positioning System (GPS)

GPS is a space based satellite navigation system. The accuracy is at best around 1-5 metres when interacting with a smartphone outdoors (Community Health Maps, 2014) and GPS radio signals are weakened by roofs, walls and other objects. Also with GPS alone it is impossible to determine which floor of a building a user is on. For these reasons GPS technology is more suited to outdoor positioning system rather than IPS.

3.3 Wi-Fi Based Positioning System (WPS)

WPS work by measuring the intensity of the received signal strength (RSS) to calculate how far they are from access points. Wireless access points are uniquely identified using the method of fingerprinting. Fingerprinting is the process of using SSID and MAC addresses of Wireless access points to uniquely identify the wireless access point. This in combination with GPS location data sent from that wireless access point is used to find the geographic location of that wireless access point. WPS systems may be combined with cell phone tower triangulation and GPS to provide reliable and accurate position data. The advantages are most smartphones support Wi-

Fi, it's cheap, wireless access points are installed in most places and the hardware lasts a long time. The problems are it requires multiple access points, the Wi-Fi hotspots database must be constantly updated to keep up with changes and it does not work if out of range of Wi-Fi signals (Zahradnik, 2013). Due to restrictions by Apple you cannot release mobile applications onto the App Store that scan wireless access points making it impossible to develop IPS using Wi-Fi for iOS (Newman, 2010). This restriction eliminates the iOS market which has a smartphone market share of 42.5% across the UK (Page, 2015). WPS are already used by many companies to locate users and objects indoors. Alone Wi-Fi based positioning systems aren't great, but when combined with other technologies including gyroscopes, accelerometers and GPS it can be very accurate as WifiSlam (Pnazarino, 2013) and SiRFusion (Francica, 2014) have proved. WifiSlam works by combining GPS and Wi-Fi signals with recorded trajectories from the smartphone including the accelerometer, gyroscope and magnetometer to calculate the users position inside a room based against a map. SiRiFusion works by combining Wi-Fi signals with the gyroscope and accelerometer to calculate the users positioning using trilateration. Overall WPS are one of the most practical solutions, but the restriction by Apple rules out nearly half of the market so they aren't viable at this moment in time.

3.4 Radio-frequency Identification (RFID)

RFID is the wireless use of electromagnetic fields to transfer data. It works by automatically identifying and tracking tags attached to objects. Each of the tags contains electronically stored information. There are different types of RFID tags including electromagnetic, interrogating radio waves and battery based (RFID Centre, 2013). RFID does not necessarily need to be within line of sight of the reader and may be embedded in the tracked object. RFID is one method for Automatic Identification and Data Capture (AIDC). RFID has its uses being instant and cheap, but it is not suitable for IPS as it is limited to just 20 cm range and it doesn't report signal strength or distances making it very hard to calculate an object's position.

3.5 Visible Light Communication (VLC)

VLC systems use visible light between 400 and 800THz to transmit data (Free Space Optics, 2011). The pulses can be controlled to work in a certain pattern. This pattern is not detectable by the human eye, but can be picked up by a camera in a smartphone. Calculations are then performed on the device to determine the user's position (Cangeloso, 2013). This technology looks very promising as it is instantaneous and very accurate. The problem is that it needs to be built into LED lighting from the start and cannot be retrofitted. With more and more people investing in LED lighting (Marcacci, 2014) this technology has huge potential in the near future, but companies need to start including it in LED lighting now. Bytelight is a company that creates

this technology, they combine LED with BLE and other inertial device sensors to transform LED lights into indoor location waypoints (Bytelight, 2014).

3.6 Ultrasound

Ultrasound is a seesaw sound pressure wave with a frequency greater than the upper limit of the human hearing range. It works by measuring how long it takes for sound to travel between transmitters in a known position to a user or object in an unknown position. The technology then uses these values to triangulate a position. There are issues with ultrasound as many objects can interfere and reflect signals and it is also not instantaneous. It's cheap and fairly accurate in a good environment, but the limitations are too great for it to be used as an IPS (Jaakko Vuorio, 2014).

3.7 Infrared

Infrared technology works by using line of sight communication between a transmitter and a receiver. The benefits are it is small and lightweight, but the disadvantages include interference with sunlight and fluorescent lighting, security issues and it is expensive (Zahid Farid, 2013). This means infrared technology would not be a suitable IPS option.

3.8 Bluetooth Low Energy (BLE)

BLE is an improvement implementation over the classic Bluetooth V2.1 and V3. Differences between Classic Bluetooth technology and Bluetooth low energy technology are summarised in Table 3 (Saltzstein, 2012):

Table 3: Classic Bluetooth vs Bluetooth low energy

	Classic Bluetooth technology	**Bluetooth low energy technology**
Data payload throughput (net)	2 Mbps	Approximately 100kbps
Robustness	Strong	Strong
Range	Up to 1000m	Up to 250m
Local system density	Strong	Strong
Large scale network	Weak	Good
Low latency	Strong	Strong
Connection set-up speed	Weak	Strong
Power consumption	Good	Very strong
Cost	Good	Strong

As already mentioned in section 2.3 the main feature of BLE is its lower power consumption that makes it possible to power a small device using a tiny coin cell battery.

3.9 Summary

Magnetic positioning looks very promising, but it takes too long to set up and is difficult to produce on a large scale. GPS isn't accurate enough indoors as roof and building materials interfere. Wi-Fi is another very exciting technology, but the limitation by Apple which restricts developers from accessing Wi-Fi RSS eliminates this from the iOS market restricting a lot of potential sales. RFID only has 20 cm range so it would not be practical. Ultrasound is not instantaneous so user's position won't be in real-time. VLC is another exciting technology, but there isn't enough LED lighting which includes the IPS technology in real-world situations yet.

Infrared is very expensive and interferes with lighting. BLE has a lot of potential and is relatively cheap, easy to set up, fast and available on most modern smartphones. BLE appears to be the most suitable technology to the applications envisaged in this project.

4 Specification Phase One: Evaluating iBeacons

The main purpose of this phase is to understand how iBeacon technology works. This will involve evaluating the operating characteristics of iBeacons to find out the most accurate, efficient and suitable way to position a user indoors. I will use these findings to help with the second stage of the project.

There are two main ways to calculate the location of an object, the empirical method and through mathematical modelling. The empirical method determines the position by matching an unknown location with a large set of known locations using k-nearest neighbour. K-nearest neighbour works based on minimum distance from the query instance to the training samples (known locations) to determine K-nearest neighbours. After the K-nearest neighbours are gathered we take simple majority of these K-nearest neighbours to be the prediction of the query instance (Teknomo, 2012). The problem with the empirical method is it requires a comprehensive on site survey and will be inaccurate if there is any significant change in the environment such as moving people or objects so it is not suitable to this kind of system. Mathematical modelling is the more popular approach to positioning the location of a user. This works by approximating signal propagation and finding angles or distance. Inverse trigonometry will then be used to determine the location. Trilateration (distance from anchors) and Triangulation (angle to anchors) use mathematical modelling. IBeacons only send out distance from anchor readings. So the only way I can determine the location of a user is through trilateration. Part of this phase will involve investigating how well iBeacons work with trilateration.

One of the main aims of this project is to deliver content to the user. The only way this can be done is triggering content when the user enters a specific distance of an iBeacon. I will evaluate how well this radius system works with multiple iBeacons.

Another method of delivering content to users could be through a zone system which is a refinement of the radius system i.e. uses multiple radii. This would involve multiple radii. Where initial content such as sound would be delivered at the outer radius and then modified at the inner radius. I will evaluate how well this zone system works with multiple iBeacons.

IBeacons include a customisable setting called advertising interval. Advertising interval is how often the iBeacon broadcasts RSSI, accuracy (received power), proximity UUID, major and minor values. IBeacon advertising intervals can range from 50 milliseconds to 2000 milliseconds

(Santos, 2014a). I will evaluate different advertising intervals and see how it impacts the battery life and performance of the iBeacons.

Estimote doesn't have any guidelines when it comes to the physical position (on the floor, ceiling etc.) of iBeacons. I will evaluate the use of iBeacons at different locations around the room and see how this impacts accuracy and performance.

According to the official Estimote guideline (Puchta, 2014b) the accuracy deviates around 5-6cm at 20cm, 15cm at 100cm and 2-3m at 10m. I will test the iBeacons to see how much the accuracy deviates in reality.

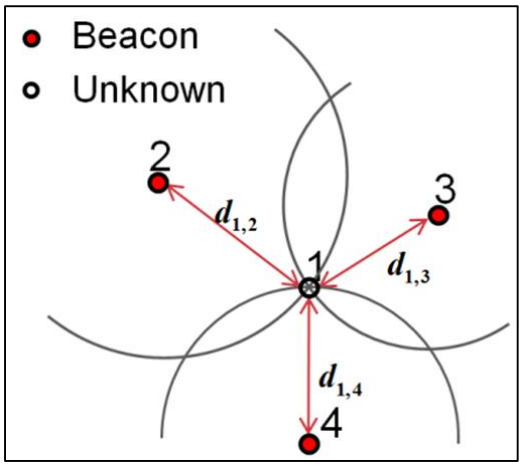
5 Design Phase One: Evaluating iBeacons

In this design section I will explain and design how I will evaluate the operating characteristics of iBeacons.

5.1 Trilateration

Trilateration is the process of determining absolute or relative locations of points by measurement of distances, using the geometry of circles, spheres or triangles. In this case I will be using circles as illustrated in Figure 2. This investigation will involve placing three iBeacons in set locations and then calculating the user's position using the distance from each of the three iBeacons. From this investigation I will be able to evaluate how well trilateration works with iBeacons and decide whether it is suitable to determine the position of a user.

Figure 2: Trilateration illustration (Nan Wu, 2011)

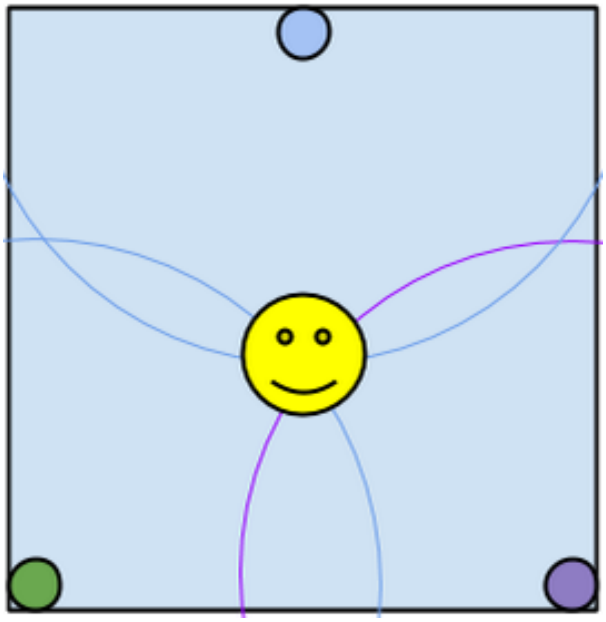


The trilateration works by using the x and y coordinates of three beacons. This combined with the distance to the three beacons from the users unknown position allows us to use each beacon as a circle using the distance as a radius and the x and y position of the beacons as the centre coordinates. The x and y values in this equation are where the beacon circle intersects with the x and y axis, the r value is the radius and the beacon.x and beacon.y values are the beacon centre coordinates. Using the equation for circles we can combine the three beacon circles to calculate the x and y coordinates of the user (Cotrell, 2012). The full implementation is mentioned in more detail in section 6.1.2.

For this algorithm to work each set of iBeacon coordinates needs to be precisely measured and then scaled to match the grid on the application. Then using the Cordova Estimote API the distance from each iBeacon can be calculated within the application.

5.1.1 Environment

Figure 3: Trilateration environment design



In the trilateration environment design shown in Figure 3 the Purple, green and blue circles represent the location of the three iBeacons. The purple, green and blue circle edges are the intersecting circles showing how far away the user is from each beacon. The yellow face marks the users position in the environment using trilateration. The test area will be 100cm x 100cm. The purple beacon will be placed in position (0,0), the blue beacon will be placed in position (0,100) and the green beacon will be placed at (50,100).

5.1.2 Mobile application

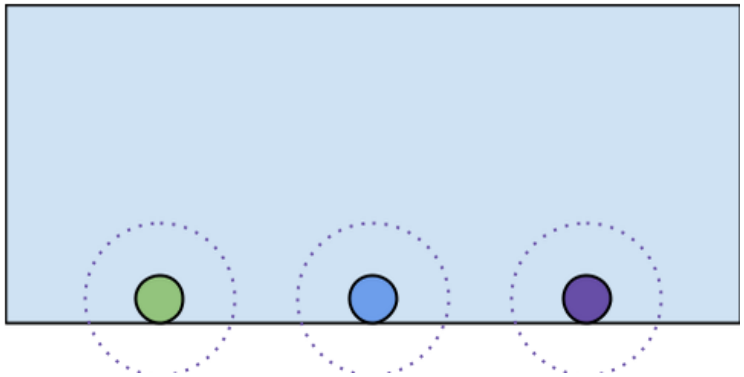
The mobile application display will mirror the environment, but use a different scale. The scale will be 2px to 1cm so the purple beacon will be in position (0, 0) the blue beacon will be placed in position (0,200) and the green beacon will be placed at (100,200).

5.2 Radius

To determine when a user enters and exits the vicinity of an iBeacon the distance from each beacon will be checked against a radius. The purpose of this test is to see if the application can accurately determine if the user enters or exits the vicinity of the iBeacon and to evaluate the performance. This will be used to deliver content to users.

5.2.1 Environment

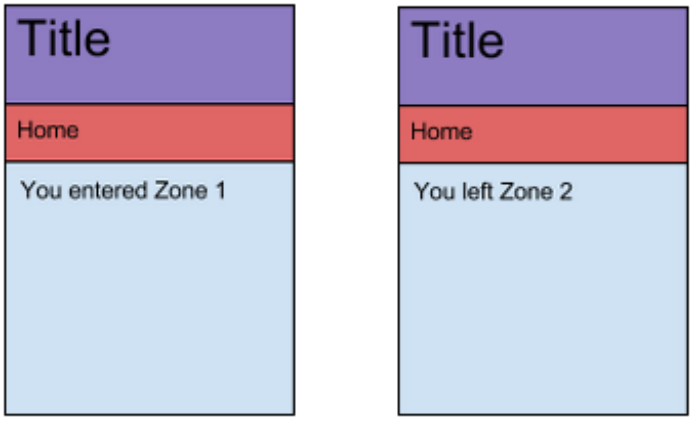
Figure 4: Radius environment design



In the radius environment design each of the three iBeacon's will be located 150cm apart and have a 50cm radius which is represented by the dotted circle edge. See Figure 4

5.2.2 Mobile application

Figure 5: Radius mobile application design



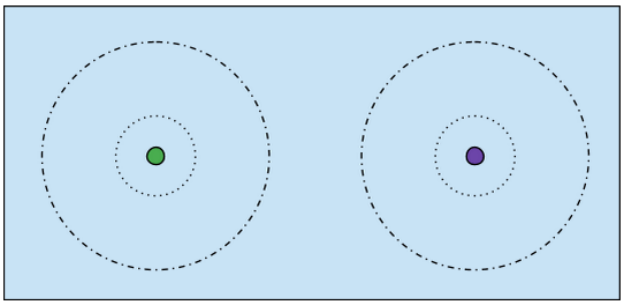
The user will be alerted when they leave or enter the 50 cm radius of the beacon through a message on the mobile application. See Figure 5.

5.3 Zones

This method is a more complex method of the radius above, as it would use more than one radius. It would work by having an outer radius and an inner radius. This test will involve two different beacons that will each trigger different audio. When a user enters the outer radius of one of the beacon's audio will start playing very quietly, the closer the user gets to inner radius the louder the audio becomes. The purpose of this test is to trigger and then adjust media volume depending on precise distance which will be needed for content delivery in the prototypes.

5.3.1 Environment

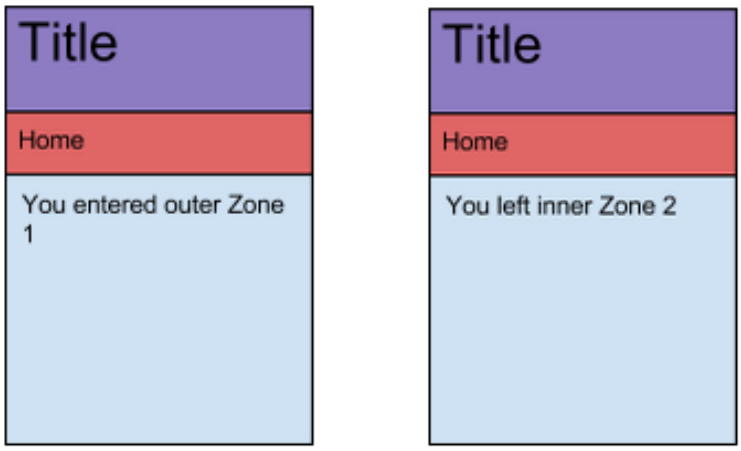
Figure 6: Zone environment design



The purple and green circles represent the iBeacons. The dotted line represents the inner radius of 50 cm. The alternative dotted/dashed line represents the outer radius of 200 cm. The beacons are located 5 metres apart. See Figure 6.

5.3.2 Mobile application

Figure 7: Zone mobile application design



When the user enters the outer radius of 200 cm it should notify the user and start playing an audio sound very quietly that should gradually get louder the closer the user gets to the beacon. When the user leaves the outer radius and enters the inner radius of 50 cm it should notify the user and turn up the volume to full. When the user completely leaves the outer radius the audio should stop and it should notify the user. There should be no conflict between the different iBeacons and audio files. See Figure 7.

5.4 Optimum advertising interval from the iBeacon

Advertising interval is how often the iBeacon broadcasts RSSI, accuracy (received power), proximity UUID, major and minor values. IBeacon advertising intervals can range from 50 milliseconds to 2000 milliseconds (Santos, 2014a). I will test the advertising intervals at 100ms,

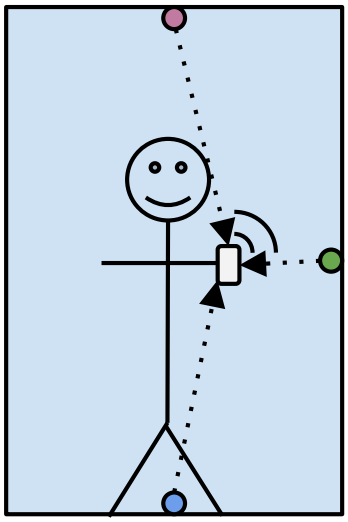
200ms, 500ms, 750ms, 1000ms, 1500ms and 2000ms. The advertising intervals of each beacon can be changed using official Estimote application. The test will involve walking in and out of beacon zones and timing how long it takes the application to pick it up. The purpose of this test is to find out the most suitable (optimum) advertising interval for this project. This will be determined by how it impacts the performance of the application, how instantaneous it is and how it affects the battery life of the iBeacons.

5.5 Optimum position to place the iBeacon

The iBeacons will need to be placed somewhere in the room, either on the floor, stuck to something or attached to the ceiling. I couldn't find documentation telling me where to position the iBeacons, but most iBeacon projects I found aligned the beacons at chest height, as this is usually where users hold their phone. I will be evaluating how the position of the iBeacon impacts the accuracy of the user's distance from the iBeacon. This procedure will involve positioning one iBeacon on the ceiling, one at chest height and one on the floor. I will then stand directly between the three beacons and evaluate how accurate the distances are and discuss any issues.

5.5.1 Environment

Figure 8: Optimum position environment design



The Purple, green and blue circles represent the location of the three iBeacons. The purple beacon is located on the ceiling, the green at chest height and the blue on the floor. I will then stand in a single location directly between the beacons as shown with the stickman and evaluate the accuracy of each beacon position. See Figure 8.

5.5.2 Mobile application

Figure 9: Optimum position mobile application design

The mobile application will need to output the beacon details so I can evaluate which position is optimum. The beacon details will include the major, minor, RSSI, distance and beacon colour. See Figure 9.

5.6 Optimum distance from the iBeacon

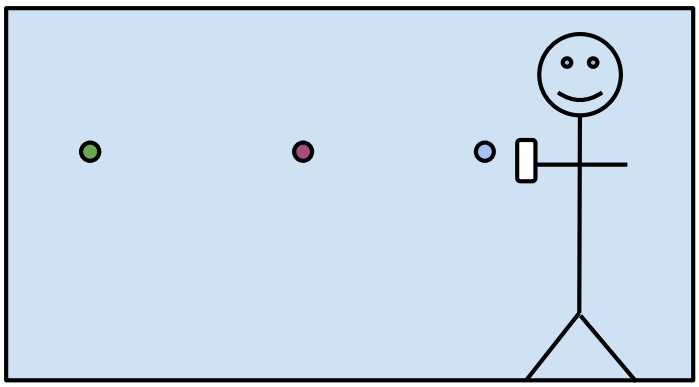
In the Estimote documentation it gives a guideline on how the iBeacon accuracy deviates at specific distances (Puchta, 2014b):

- Distance of 20cm has a deviation of 5-6cm
- Distance of 1m has a deviation of 15cm
- Distance of 10m has a deviation of 2-3m

The purpose of this investigation is to find out the best distances to place a radius when delivering content to users.

5.6.1 Environment

Figure 10: Optimum distance environment design



I will stick the iBeacons to a wall at specific distances. The green, purple and blue circles represent the different beacon locations. The stick man represents my position which will be specific distances away from each beacon. The blue beacon will be 20cm away, the purple beacon 100cm away and the green beacon 200cm away from my position. See Figure 10.

5.6.2 Mobile application

Figure 11: Optimum distance mobile application design

The mobile application will need to output the beacon details so I can evaluate each of the beacon distances in Figure 11. The beacon details will include the major, minor, RSSI, distance and beacon colour.

6 Implementation Phase One: Evaluating iBeacons

Here I will talk about how I implemented and set up each of the prototypes so I could evaluate the operating characteristics of the iBeacons.

6.1 Trilateration

In addition to the original 100cm x 100cm test I decided to carry out a more complex test on a bigger space of 300cm x 400cm. I did this as this is the sort of area Yellobrick would use for the application. The scale of the second system was 1cm in in physical space to 1px on the iPhone screen display.

6.1.1 Environment

Setting up the environment was fairly simple. For the 100cm x 100cm grid system I drew a grid on a piece of plywood drawing a line every 10cm vertically and horizontally. I then positioned the iBeacons at positions (0,0), (0,100) and (50,100). For the bigger space of 300cm x 400cm I used a laser measuring device to measure the size of the space. I then positioned the beacons around the space at positions (0,0), (150, 400) and (300,0). I have taken pictures of the two environments shown in Figure 12 and Figure 13. The dotted red circles highlight the positions of the iBeacons.

Figure 12: Trilateration 100cm x 100cm environment picture

Figure 13: Trilateration 300cm x 400cm environment pictures

6.1.2 Mobile application

I wrote a JavaScript function to perform the trilateration calculation using Kdzwinel's implementation from GitHub as a guide (kdzwinel, 2014). See Code listing 1.

Code listing 1: Trilateration function for 300cm x 400cm environment

```
function triangulation(array) //the array stores the distance to each beacon
{
    for (var i = 0; i < array.length; i++)  //loops over the array grabbing the distance to each beacon
    {
        if(array[i][1] === 1){var dA = parseInt(array[i][4])}
        if(array[i][1] === 2){var dB = parseInt(array[i][4])}
        if(array[i][1] === 3){var dC = parseInt(array[i][4])}
    }

    //room size in scale (1cm = 1px)
    var roomWidth = 300; //in cm
    var roomLength = 400; //in cm
    var beaconCoordinates = [[5,5], [145,395], [295,5]]; //position of each beacon in scale

    //create a variable for each of the x and y coordinates of each beacon
    var aX = parseInt(beaconCoordinates[0][0]);
    var aY = parseInt(beaconCoordinates[0][1]);
    var bX = parseInt(beaconCoordinates[1][0]);
    var bY = parseInt(beaconCoordinates[1][1]);
    var cX = parseInt(beaconCoordinates[2][0]);
    var cY = parseInt(beaconCoordinates[2][1]);

    //trilateration formula
    var S = parseInt((Math.pow(cX, 2.) - Math.pow(bX, 2.) + Math.pow(cY, 2.) - Math.pow(bY, 2.) + Math.pow(dB, 2.) - Math.pow(dC, 2.)) / 2.0);
    var T = parseInt((Math.pow(aX, 2.) - Math.pow(bX, 2.) + Math.pow(aY, 2.) - Math.pow(bY, 2.) + Math.pow(dB, 2.) - Math.pow(dA, 2.)) / 2.0);
    var y = ((T * (bX - cX)) - (S * (bX - aX))) / (((aY - bY) * (bX - cX)) - ((cY - bY) * (bX - aX)));
    var x = ((y * (aY - bY)) - T) / (bX - aX);

    //x and y position of user
    var position = [x,y];
}
```

First the code loops over the distances array which is parsed through the function to get the distance to each iBeacon. This distance array holds the information about each of the iBeacons, more details about how the data is pushed to the array can be found in section 6.5.2 later in

this report. Then the environment values roomWidth, roomLength and beaconCoordinates are all hardcoded into the system to match the environment. I then store each of the beacon coordinates as variables which are then used in combination with the distance to each beacon variables to calculate the x and y position of the user.

To display the trilateration system on the mobile application I used Scalable Vector Graphics (SVG) to draw the beacons and user position. See Code listing 2.

Code listing 2: Trilateration SVG elements

```
<svg width="300" height="400" class="svg-map">
    <g transform="translate(0,400)">
        <g transform="scale(1,-1)">
        <circle class="boundary blueberry" id="blue-bound" cx="5" cy="5" r="0"/></circle>
        <circle class="boundary mint" id="mint-bound" cx="145" cy="395" r="0"/></circle>
        <circle class="boundary ice" id="ice-bound" cx="295" cy="5" r="0"/></circle>
        <circle class="blueberry beacon-map" cx="5" cy="5" r="5"/></circle>
        <circle class="mint beacon-map" cx="145" cy="395" r="5"/></circle>
        <circle class="ice beacon-map" cx="295" cy="5" r="5"/></circle>
        <rect id="marker" x="95" y="95" width="10" height="10"/></rect>
        </g>
    </g>
</svg>
```

Then using different variables from the trilateration formula I updated the SVG so it matched the real time data. To update the circle edges I changed the “r” (radius) value of the “blue-bound”, “mint-bound” and “ice-bound” to the distance to each beacon respectively taking into consideration the ratio.

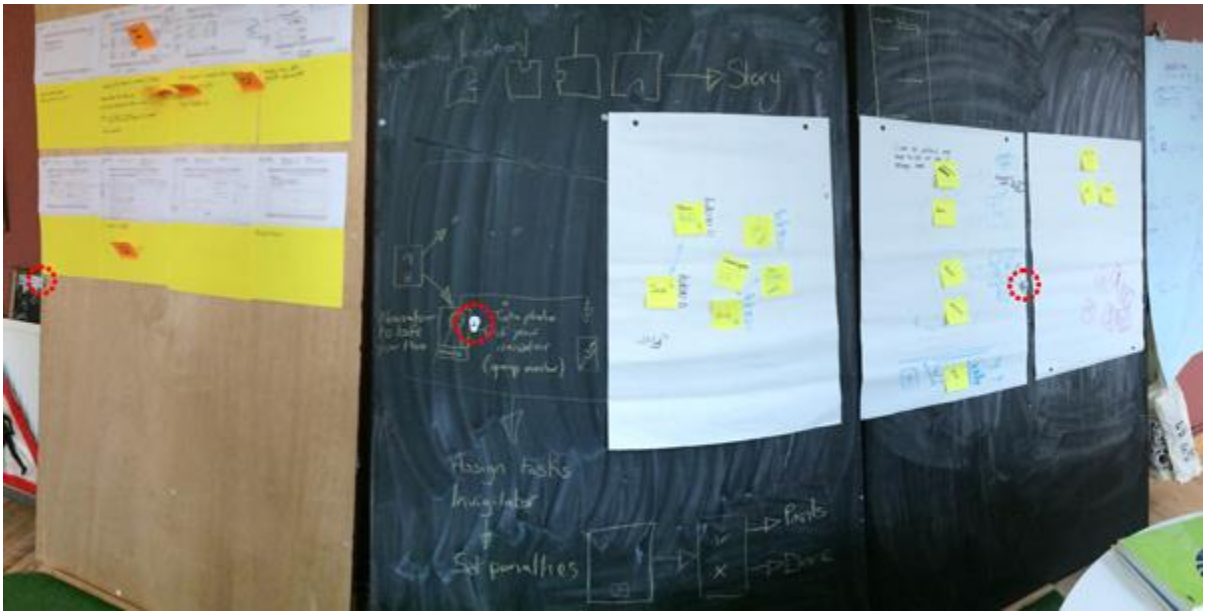
To update the position of the “marker” (user) I updated the rectangles x and y values to use the new calculated x and y values from the trilateration formula.

6.2 Radius

6.2.1 Environment

I stuck the iBeacons using Blu-tack to a piece of plywood. I positioned them 150cm apart using a tape measure. I have taken a picture of the iBeacons positioned on the plywood. See Figure 14.

Figure 14: Radius environment picture



6.2.2 Mobile application

To determine when a user enters and exits the vicinity of an iBeacon I wrote a function that will loop over each of the iBeacons. See Code listing 3. The function will then check the beacon major, minor, distance and beaconVariable. First it will check if it is the relevant iBeacon using the major and minor. Then it will check if the user has entered the 50cm radius. If the user is outside the radius the beaconVariable is set to 0. If the user enters the radius the beaconVariable is set to 1. This is used to determine if the user has just entered or exited the iBeacon.

Code listing 3: Radius function for iBeacon1

```
var beacon1 = 0;  //beacon1 status is set to not visited by default
    for( var i=0; i<array.length; i++ ) //loops over each of the iBeacons
    {
    if(array[i].minor === 1 && array[i].major === 44333) //check if it's the correct iBeacon using major and minor
    {
        var distance = parseInt(array[i].distance); //get the distance value
        if(distance <= 50) //check if the distance if inside the 50cm radius
        {
            if(beacon1 === 0) //check if the user has just entered from outside
            {
                $('#story-hint').text("You entered Zone 1"); //notify the user they have entered the zone
            }
            beacon1 = 1; //set the beaconVariable to true as they are inside the radius
        }
        else //if they are not inside the 50cm radius
        {
            if(beacon1 === 1) //check if the user has just left the inside radius
            {
                $('#story-hint').text("You left Zone 1");  //notify the user they left the zone
            }
            beacon1 = 0; //set the beaconVariable to false as they are inside the radius
        }
    }
}
```

6.3 Zones

6.3.1 Environment

I positioned the iBeacons 5 metres apart using a tape measure. I then used Blu tack to attach the iBeacons to a piece of plywood. Refer to first 2 iBeacons in Figure 14

6.3.2 Mobile application

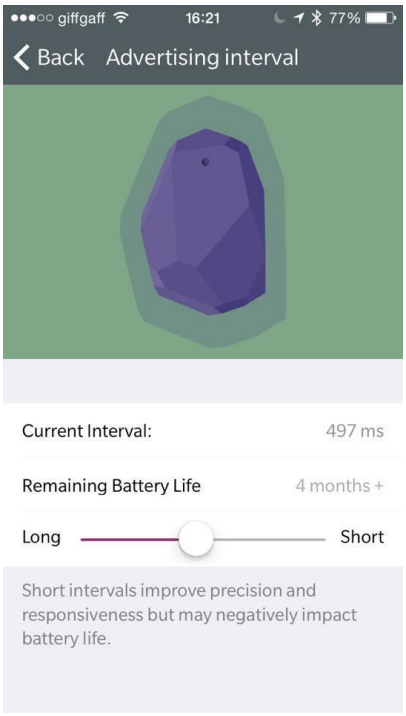
I used the radius function as a basis for developing the zone function. See Code listing 4. The radius function already included one 50cm radius so all I had to do was add a few lines of code to play the audio at full volume when they entered the inner radius. For the outer radius I checked they were inside the radius of 200cm and outside the radius of 50cm. I then calculated a percentage to find out how close the user was to the iBeacon and then used this to adjust the audio volume. This would also require an additional variable so I could detect when the user enters and exits the different zones.

Code listing 4: Zone function for iBeacon2

```
var zone2a = 0; //each of the status' are set to 0 by default
var zone2b = 0;
var play2 = 0;

if(minor === 2 && major === 44333)
{

    if(distance <= 200) //this checks if the user has entered the outer radius zone
    {
        if(play2 === 0) //check if the audio is playing
        {
            play2 = 1; //set the audio playing to true
            media2.play(); //plays the audio for beacon 2
        }

    }

    if(distance <= 50 && zone2a === 0) //if the user enters the inner zone and wasn't previously inside the inner zone
    {
        $('#story-hint').text("Inner Zone 2"); //notify the user they are in the inner zone
        $('#story-text').css("font-size", "35px");
        zone2a = 1; //set inner zone to true
        zone2b = 0; //set outer zone to false
        media2.setVolume(1.0); //set the audio volume to full
    }

    else if(distance <= 200 && distance > 50) //if the user is inside the outer zone and outside the inner zone
    {
        if(zone2b === 0) //check is user was previously in the outerzone
        {
            $('#story-hint').text("Outer Zone 2"); //notify the user they are in the outer zone
            $('#story-text').css("font-size", "25px");
            zone2b = 1; //set outer zone to true
            zone2a = 0; //set inner zone to false
        }

        var per = distance/200; //calculate the percentage of the distance to the beacon
        var vol = 1-per;
        media2.setVolume(vol); //set the volume to the percentage

    }

    else if(distance > 200) //if the user is outside the outer zone and was previously inside it
    {
        $('#story-hint').text("YOU LEFT ZONE 22222"); //notify the user they left the zone
        $('#story-text').text("Searching..");
        $('#story-text').css("font-size", "15px");
        zone2a = 0; //set innter zone to false
        zone2b = 0; //set outer zone to false
        media2.stop(); //stop the audio
    }
}
```

6.4 Optimum advertising interval from the iBeacon

Figure 15: Changing the advertising interval using the official Estimote mobile application



To change the advertising interval I used the official Estimote application. See Figure 15. The official Estimote application is available on the App Store (Estimote, 2014). I searched devices and then clicked the iBeacon I wanted to change. I then had to login and verify I owned the iBeacon. Once I had done this I could change the advertising interval to a value between 100ms and 2000ms. The values I changed the advertising interval to were 100ms, 200ms, 500ms, 750ms, 1000ms, 1500ms and 2000ms.

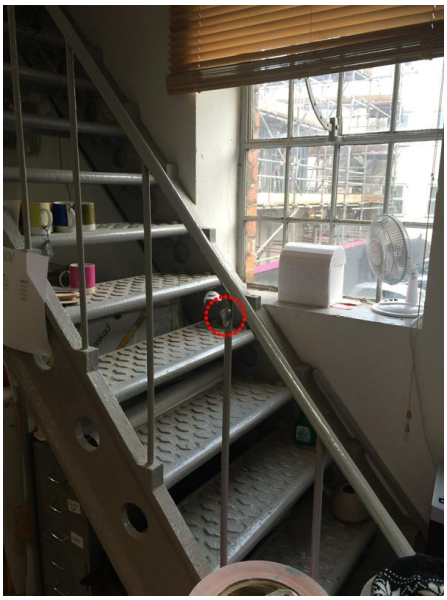
I then ran the zone mobile application I developed previously in 6.3.2 and timed how long it took for the application to realise I entered and exited the iBeacon radius.

6.5 Optimum position to place the iBeacon

6.5.1 Environment

To attach the iBeacons onto the ceiling and at chest height I used Blu-tack. I then positioned the other beacon on the floor. Figure 16 shows pictures of the iBeacon at chest height (on the stairs) and the iBeacon on the floor.

Figure 16: Optimum position environment pictures



6.5.2 Mobile application

To display the major, minor, RSSI, distance and colour details of the iBeacon I first needed to loop over each iBeacon and store the data in an array using JavaScript. See Code listing 5. This array is used to store the different iBeacon details.

Code listing 5: Store iBeacon data function

```
$.each(beaconInfo.beacons, function(key, beacon)
{
    //convert colour code to colour name
    if (beacon.color === 3 ){var col = 'blueberry'}
    else if (beacon.color === 1 ){var col = 'mint'}
    else if (beacon.color === 2 ){var col = 'ice'}
    else {var col = 'n/a'}

    //add data for each beacon to an array object
    var tempArray = {major:beacon.major, minor:beacon.minor, rssi:beacon.rssi, distance:formatDistance(beacon.distance), colour:col};

    //push array of objects to the array
    array.push(tempArray);
});
```

I then made this iBeacon information visible to the user by looping over the array and appending the beacon information to the page-story div using JQuery. See Code listing 6.

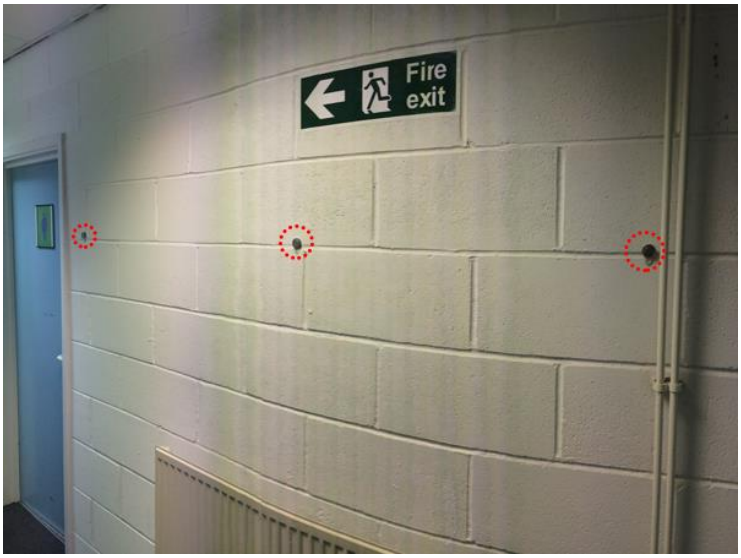
Code listing 6: Display iBeacon information

```
for( var i=0; i<array.length; i++ )
{
    $("#page-story").append("<p id='console"+i+"'>major: "+array[i].major+", minor: "+array[i].minor+"
        , rssi: "+array[i].rssi+", distance: "+array[i].distance+", colour: "+array[i].colour+"</p>");
}
```

6.6 Optimum distance from the iBeacon

6.6.1 Environment

Figure 17: Optimum distance environment picture



To position iBeacons at specific distances I measured distances along a wall using a ruler and then stuck the iBeacons at 20cm, 100cm and 200cm away from the starting position using Blu-tack. See Figure 17. The starting position is where I would stand to test the deviation of the distances. The picture shows the iBeacons positioned on the wall.

6.6.2 Mobile application

I ran the zone mobile application from 6.3.2 and compared the distance values for each iBeacon to the actual distance I measured using a ruler.

7 Evaluation Phase One: Evaluating iBeacons

Here I will discuss how each of the tests performed and what I learnt.

7.1 Trilateration

The original test using the 100cm x 100cm environment was fairly accurate with the position only deviating by 10-20 cm. This though didn't tell me much as the test area was far too small and was unlikely to be used in the real world so I decided to focus the test on a larger 300cm x 400cm environment.

Figure 18: Trilateration mobile application using the 300cm x 400cm environment

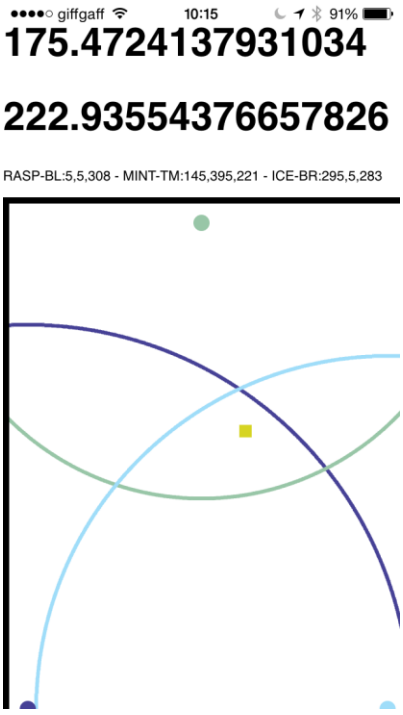


Figure 18 shows screenshots of the trilateration mobile application in the 300cm x 400cm environment. The Blueberry, Mint and Ice coloured circles are the beacons in their real life positions and the coloured circle edges show the distance measured from each beacon. The yellow square is the user's position.

The user's x and y position was determined using the trilateration function and then drawn onto the mobile application. For this to be accurate the three circle edges would all touch in one place and be relevant to my position in real space which was directly between the 3 iBeacons. Note the solution method used finds the centre of the overlapping region (Cotrell, 2012). As you can see it is far from accurate as the circles don't all intersect at one point like in Figure 3. It was also very jumpy with the distance.

I carried out research to establish what was causing the high level of inaccuracy. I found a few posts online regarding objects interfering with the BLE signals causing bad accuracy. This resulted in me investigating the different materials that could potentially interfere with the BLE signal. Estimote categorised interferences into these four levels (Santos, 2014b):

- Low interference potentials: wood, synthetic materials, and glass

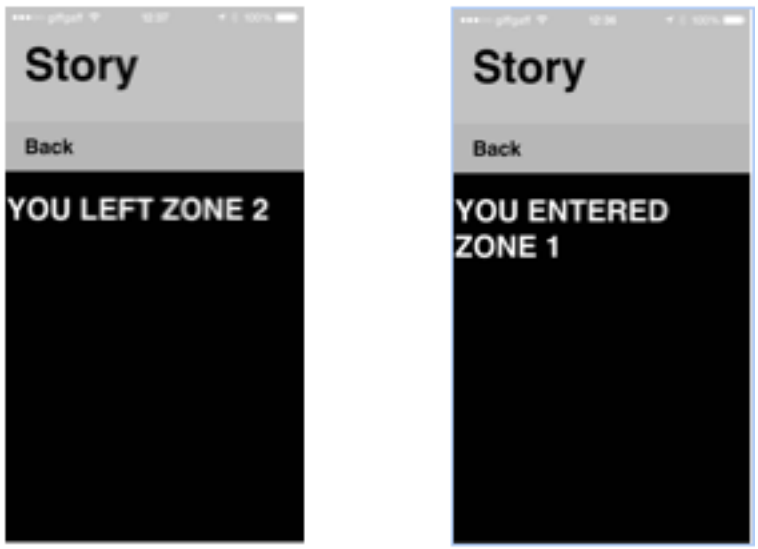
- Medium interference potentials: bricks and marble
- High interferences potentials: plaster, concrete, and bulletproof glass
- Very high interference potentials: metal, water

My assumption was that the human body caused this interference, as the room was open with none of the materials above interfering. I then repeated the test but by holding my smartphone out in front of me so my body (water) wasn't blocking any of the BLE signals. This improved the accuracy a little, but not by much. After investigating the optimum distance (discussed below) I realised that being around two metres away from the iBeacons combined with the interferences from the environment caused the accuracy problem. The best-case scenario would be the user being 200 cm away from each beacon which would still cause around 30-50 cm deviation.

7.2 Radius

The radius functionality was very accurate in terms of notifying the user at the 50cm distance as the user was notified as soon as the iBeacon distance hit 50cm. There were noticeable lag issues of approximately 1 second that resulted in the user not being notified instantly. This is caused by the phones processing power. This will be discussed in more detail when investigating the optimum advertising interval.

Figure 19: Screenshots of the radius mobile application



The screenshots in Figure 19 are a result of me leaving the radius of beacon 2 and entering the radius of beacon 1.

7.3 Zones

Figure 20: Screenshots of the zone mobile application

Outer zone 2

Center zone 1

Using zones was very effective and worked quite well as the user was notified when entering the different zones at the actual distances. There was still a small issue of lag when triggering and updating the audio volume, but this is because of the phone processor so there is nothing I can do about this. At normal walking pace the outer radius played the audio almost instantly, but when updating the audio volume there was a small delay of approximately 1 second. Then when I left the first beacon and moved into the second beacon's outer radius there was a slight delay in stopping the first audio and playing the second audio. If you walked slowly there was no overlap and the system worked fine. These screenshots in Figure 20 are a result of me leaving the outer zone of beacon 2 (200cm) and entering the inner zone (50cm) of beacon 1.

7.4 Optimum advertising interval from the iBeacon

There were noticeable lag issues with the application whatever advertising interval I used. I spoke to a few people at Estimote about this issue and they said it was because of the phones processing power. So in the future when smartphone processors are improved this issue will not exist.

Table 4: Advertising interval test results

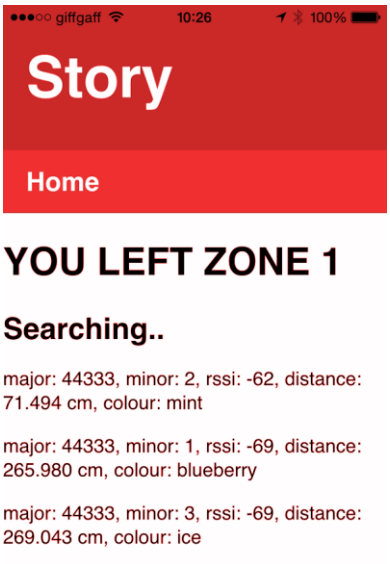
Advertising interval	Expected battery life	Comments	Response Time (approximately)
100ms	2 months+	Too many values are being processed so there is lag. Very bad battery life. Not useful for this project.	1-2 seconds
200ms	4 months+	Too many values are being processed so there is lag. Bad battery life. Not useful for this project.	1-2 seconds
500ms	9 months+	Some lag is noticeable. Decent battery life. Depending on content being delivered this could be useful.	1 second
750ms	13 months+	Very slight lag. Good battery life. Depending on content being delivered this could be useful.	< 1 second
1000ms	18 months+	No lag issues. Very good battery life. Depending on content being delivered this could be useful.	< 0.5 seconds
1500ms	24 months+	Too slow for this project. There are no lag issues regarding data values. Excellent battery life.	< 0.5 seconds
2000ms	36 months+	Too slow for this project. There are no lag issues regarding data values. Best possible battery life.	< 0.5 seconds

Based on the results displayed in Table 4, the optimum advertising intervals are between 500 and 1000ms. I will set the beacons to 750ms for the project.

7.5 Optimum position to place the iBeacon

Placing one iBeacon on the floor and attaching one to the ceiling added an additional 1-2 metres to the distance. This alone caused more inaccuracy because of the distance. There was also more interference with the floor and ceiling positioned iBeacons as my torso got in the way which contains water and causes very high interference with the BLE signals, this is mentioned in more detail in 6.1.2. Placing the iBeacon at chest height worked out to be the best position as this resulted in less interference and distance between the user and iBeacon. When I faced away from the iBeacon I encountered a problem with interference as my torso once again got in the way.

Figure 21: Screenshot of the optimum position mobile application

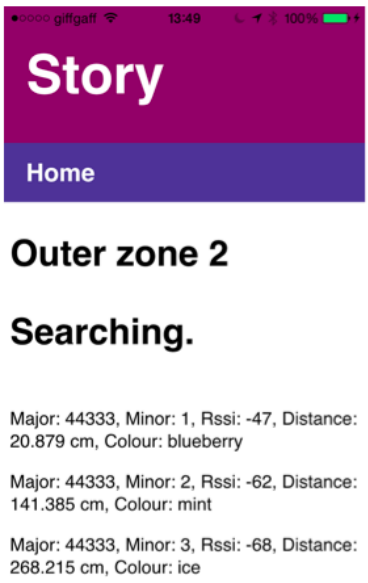


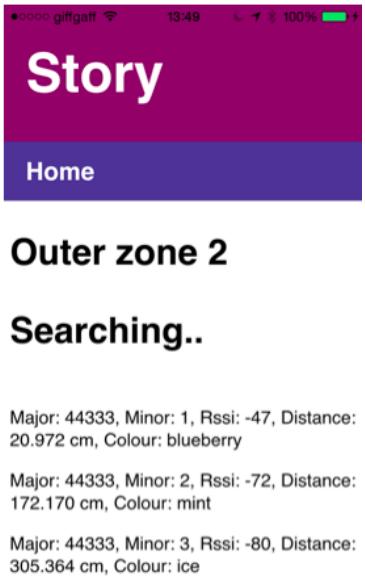
As you can see from the screenshot in Figure 21, beacon 2 has the lowest distance so the most suitable place to position the iBeacon was chest height at around 150 cm.

7.6 Optimum distance from the iBeacon

I experienced worse deviation than listed in the documentation. I think the environment I was in may have caused some of the BLE signals to reflect. The general trend was the further away the beacon was the far more inaccurate it became.

Figure 22: Screenshots of the optimum distance mobile application





I was standing approximately 20 cm away from beacon 1, 100 cm from beacon 2 and 200 cm from beacon 3. As you can see in Figure 22 the accuracy of the distance vastly increases the further away I move from the beacons. From this investigation the optimum distance from the beacon is within 1 metres and at most 2 metres otherwise it deviates too much.

8 Conclusion Phase One: Evaluating iBeacons

Here I will summaries what I discovered in phase one.

After evaluating these operating characteristics of iBeacons I now have a much better understanding of how iBeacon technology works. I am also in a much better position to develop the Yellobrick system as I understand the limitations of the technology and know how to use it properly.

From the trilateration investigation I found out that the use of trilateration is not very suitable to determining a user's position when using iBeacons, as it is not accurate enough as too many objects interfere. As a result of this investigation I will not be using trilateration in the Yellobrick system.

The radius investigation went very well as the iBeacon system accurately notified the user when they entered the 50cm radius of the iBeacon. The only issue was the lag, but this it to do with the smartphone's processor so any IPS technology using a smartphone would experience the same issue. I will use this radius method as the main condition to delivery of content and monitor when users enter and exit the vicinity of an iBeacon.

The zone investigation worked very well too. Minus the smartphone lag issues it was great as the audio played at the outer zone very quietly, then got louder as the user got to the iBeacon and the volume was set to full when the user entered the inner radius. I will use the zone method in addition to the radius method depending on the type of content that needs to be delivered to the user.

From the advertising interval investigation I learnt that the optimum advertising intervals are between 500 and 1000ms. For this project I will set the beacons to 750ms for the project.

The best place to position the iBeacons based on the optimum position evaluation was at chest height. This resulted in the least interference and highest accuracy.

When testing the accuracy of the iBeacons, the deviation for me was a lot higher than mentioned in the Estimote guideline. The general trend was further away the iBeacon the less accurate the reading and over 200cm the iBeacons were too inaccurate to use. From this investigation the optimum distance is anywhere up to one or two metres as any further away the accuracy decreases massively.

For the Yellobrick system I will not use trilateration for anything as it was too inaccurate. I will use both the radius and zone systems to deliver content and track where the users are. I will set the iBeacons advertising interval to 750ms. When positioning the iBeacons I will place them at chest height and set the radius at a maximum of 200cm away.

9 Specification Phase Two: Yellobrick Prototype

The main purpose of this second phase is to develop a user-friendly prototype for Yellobrick using my findings from phase one where I evaluated iBeacons. This prototype will allow the client to appreciate the use of iBeacon indoor positioning technology.

The Yellobrick system will be split into two sub-systems. A mobile application which will be used by Yellobrick customers and a system management interface which will be used by Yellobrick employees.

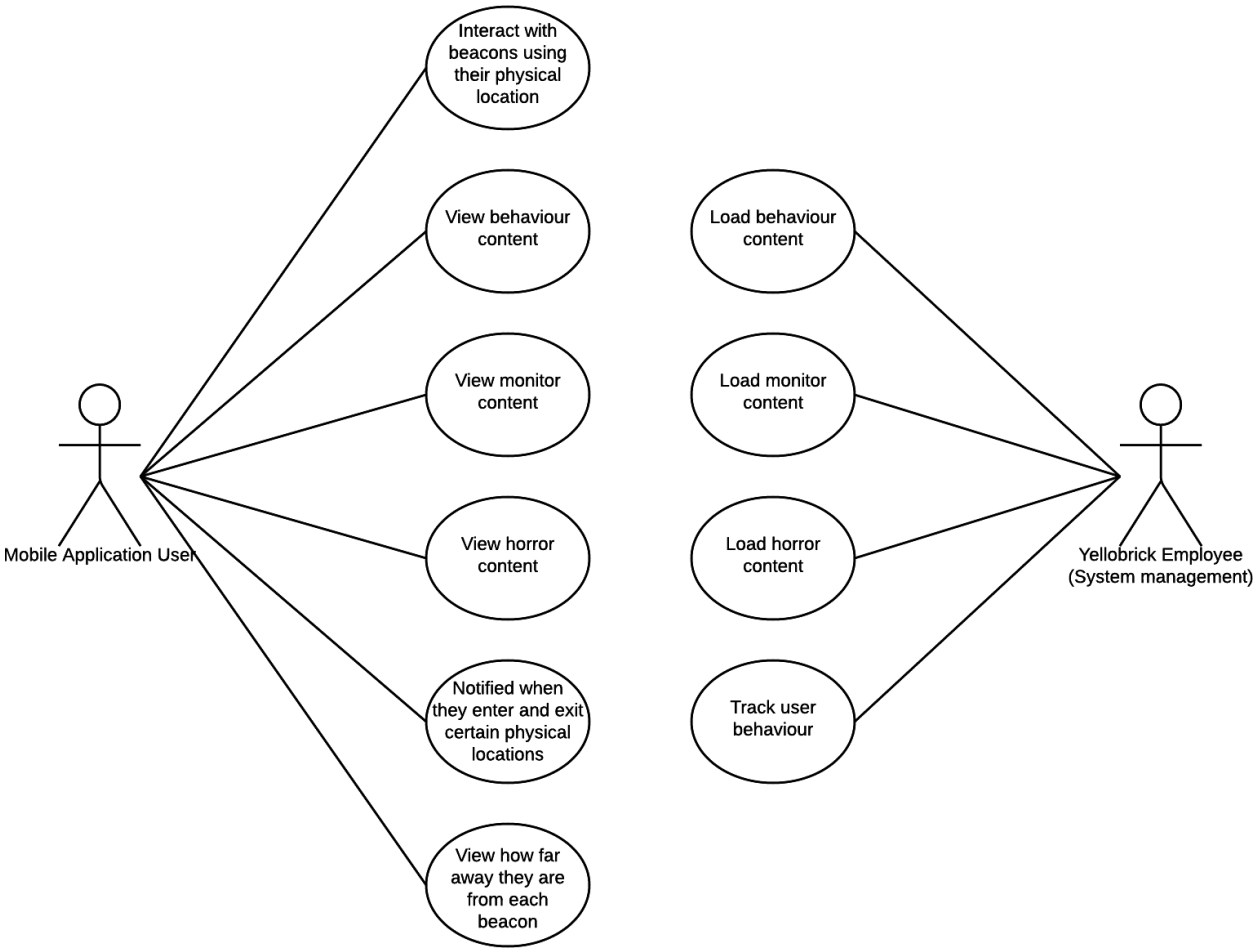
The mobile application needs to interact with the iBeacons to accurately calculate the user's position in real-time. If a user then enters a specific distance of an iBeacon the user needs to be notified via the application and the user data will need to be sent to a server efficiently. Audio content will need to be output through the mobile application to make the experience more interactive. When the user leaves the specific distance of an iBeacon the user will also need to be notified and again the data will need to be sent to a server efficiently. The mobile application needs to be user-friendly ensuring it is simple, responsive, consistent, intuitive and forgiving. It needs to be compatible with iOS and desirably Android.

The system management interface will allow Yellobrick employees to both view user behaviour anonymously and set up content on computer monitors which will trigger when user's enter specific locations. The view user behaviour section should allow the Yellobrick employee to view the complete history of the system. This means all users enter and exit times for each beacon will need to be stored on a server which should be reliable and efficient. The employee will hopefully also be able to set up different types of content on larger devices such as monitors to give the user a more interactive experience with the environment. The content will include a monitor system that displays video content for the relevant iBeacon when users are inside the iBeacon radius. This will enhance the users experience with the system making them feel more involved with the environment. Another content system will monitor behaviour of individual users which will only deliver content if an individual user visits the iBeacons in a specific order. The purpose of this is to test out a story-based game, where they can only unlock content if they have visited previous content. Another content system will involve a horror frame which only activates when one user is close to an iBeacon, if there are zero or more than one users inside the radius it wont activate. The purpose of this is to scare the user in a dark, horror story type setting. The system management interface also needs to be simple, responsive, consistent intuitive and forgiving. The interface also needs to be compatible across all major browsers.

9.1 Use case diagram

To model the interactions each of the users will have with the system I have used a use case diagram. See Figure 23. There are will be two actors, the Yellobrick employee and application user. The Yellobrick employee is someone who works for Yellobrick who can track application users through a system management interface. The Yellobrick employee will also be able to load content onto screens and monitors. The application user is someone who uses the mobile application and interacts with the iBeacons. They should also be able to view content that has been loaded onto screens and monitor by the Yellobrick employee.

Figure 23: Use case diagram



From the specification and use case diagram I have outlined the functional and non-functional requirements that will be implemented in this system. For each of the requirements I will provide a description and acceptance criteria.

9.2 Functional Requirements with acceptance criteria

Requirement: Deliver content to the user through the mobile application when their physical location is within a certain distance of the iBeacon.

- Description: The mobile application must deliver content to the user when certain conditions are met. These conditions would include entering the radius of an iBeacon.
- Acceptance criteria: When the user enters a two metre radius of an iBeacon the user should be notified via the application.

Requirement: Deliver content to the user through the system management interface when users physical locations are within a certain distance of the iBeacon.

- Description: The system management interface must also be used to deliver content to users. The content would be delivered when a user enters the radius of an iBeacon, enters the radius of the iBeacons in a specific order or when a specific number of users are inside the radius.
- Acceptance criteria:
    - When the user enters a two metre radius of an iBeacon content should be displayed on a monitor through the system management interface.
    - Content for beacon two is only displayed after the user enters the beacon one radius and content for beacon three is only displayed after the user enters the beacon one and two radius.
    - The content only plays if one user is inside the radius. If more than one user is inside the radius it should reset as if there were none.

Requirement: Each user must be uniquely identified anonymously.

- Description: To avoid privacy issues regarding the device UUID this will need to be changed into an anonymous value which can still identify each user uniquely.
- Acceptance criteria: Each user can be uniquely identified without using their device UUID.

Requirement: User data must be sent to a remote server when they enter or exit a specific radius of an iBeacon and have access to the internet.

- Description: User data including Beacon ID, User ID, Location status, date and time needs to be sent to the server if a network is available. If there is no network connection the data will need to be stored locally on the device and sent to the server when a network connection is next available.
- Acceptance criteria:
    - Beacon ID, User ID, Location status, date and time is sent to the remote server when the user enters or exits a specific radius of an iBeacon.
    - The mobile application stores user data in local storage when there is no network connection available. The data is then sent to the remote database.

Requirement: Yellobrick staff must be able to track user “dwell times” of all users.

- Description: The system management interface must show all users behaviour history which can be used to track where users have been, how long they spend at each location and the order they visited each location.
- Acceptance criteria: User data including Beacon ID, User ID, Location status, date and time is available to view on the system management interface.

Requirement: Data must be stored on a remote server in a database so the mobile application and system management interface uses the same data.

- Description: A database will need to be set up on a remote system so the user behaviour data inside the mobile application can be used by system management interface.
- Acceptance criteria: A database has been set up and stores important user behaviour data including BeaconID, UUID, Timestamp and Status.

9.3 Non-Functional Requirements with acceptance criteria

Requirement: Performance

- Description: The location of the user must be calculated very quickly so the system can deliver accurate content.
- Acceptance criteria:
    - The system needs to calculate the users current position and deliver location details via the mobile application within an acceptable time frame of up to 3 seconds.
    - The system needs to push data to the database stored on the server within an acceptable time frame of up to 3 seconds if a network connection is available.

Requirement: Efficient

- Description: Data will be sent from the mobile device to the database over the internet so it will need to be sent as efficiently as possible. To do this the least amount of data possible will be sent to the database. Data will also be exchanged between the system management interface and the database which will also need to be efficient.
- Acceptance criteria:
    - Each data item sent from the mobile application to the server will need to be justified and used in the system management interface.
    - Each data item sent from the database to the system management interface will need to be justified.

Requirement: User-friendly

- Description: The system should be simple, forgiving, responsive, intuitive and consistent. A user without any technical knowledge should be able to use, understand and never get stuck using the system. I will use the System Usability Scale (SUS) (Brooke, 1986) to measure the systems usability.
- Acceptance criteria: The SUS score of 68% is considered above average. So a score over 68% will be needed for this requirement to be deemed a success.

Requirement: Compatibility

- Description: It is an essential requirement that the mobile application works on the iOS platform and a desirable requirement that the application works on the Android platform.
- Acceptance criteria:
    - All the mobile application functionalities work on iOS mobile devices.
    - All the mobile application functionalities work on Android mobile devices (desirable)

Requirement: Portability

- Description: The system management interface needs to work across all the major browsers. This includes Chrome, Internet Explorer, Safari and Firefox.
- Acceptance criteria: The system management interface functionalities work on:
    - Chrome
    - Internet Explorer
    - Safari
    - Firefox

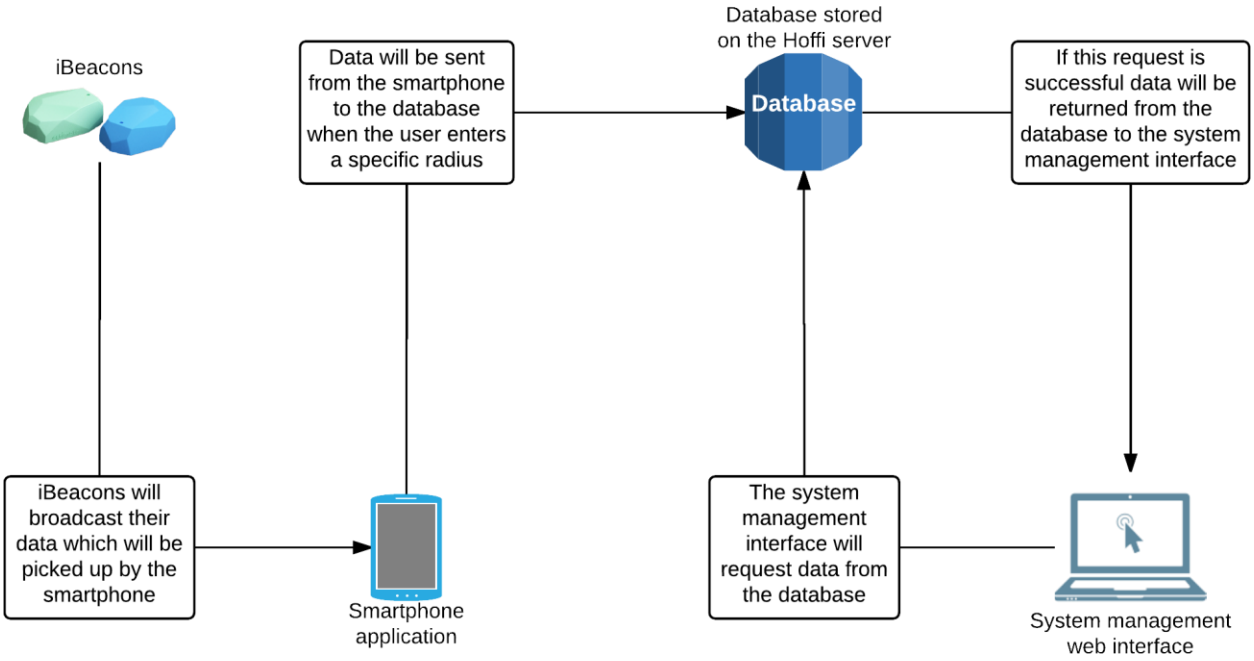
Requirement: Reliability

- Description: The system needs to accurately locate users. It is important the system delivers content only when the user is in specific areas of the room.
- Acceptance criteria: The system only delivers content when the user enters within a specific radius of an object.

10 Design Phase Two: Yellobrick Prototype

Here I will design the Yellobrick system using the functional and non-functional requirements. This system will be split into two main sections. There will be a mobile application which will be used by the users and a system management interface which will be used by the Yellobrick employees. There will also be a database which will be used to store data between the mobile application and system management interface. Figure 24 illustrates the proposed system. A recommended environment will be designed so future users can replicate the system accurately.

Figure 24: Overview of the Yellobrick system



10.1 Mobile application

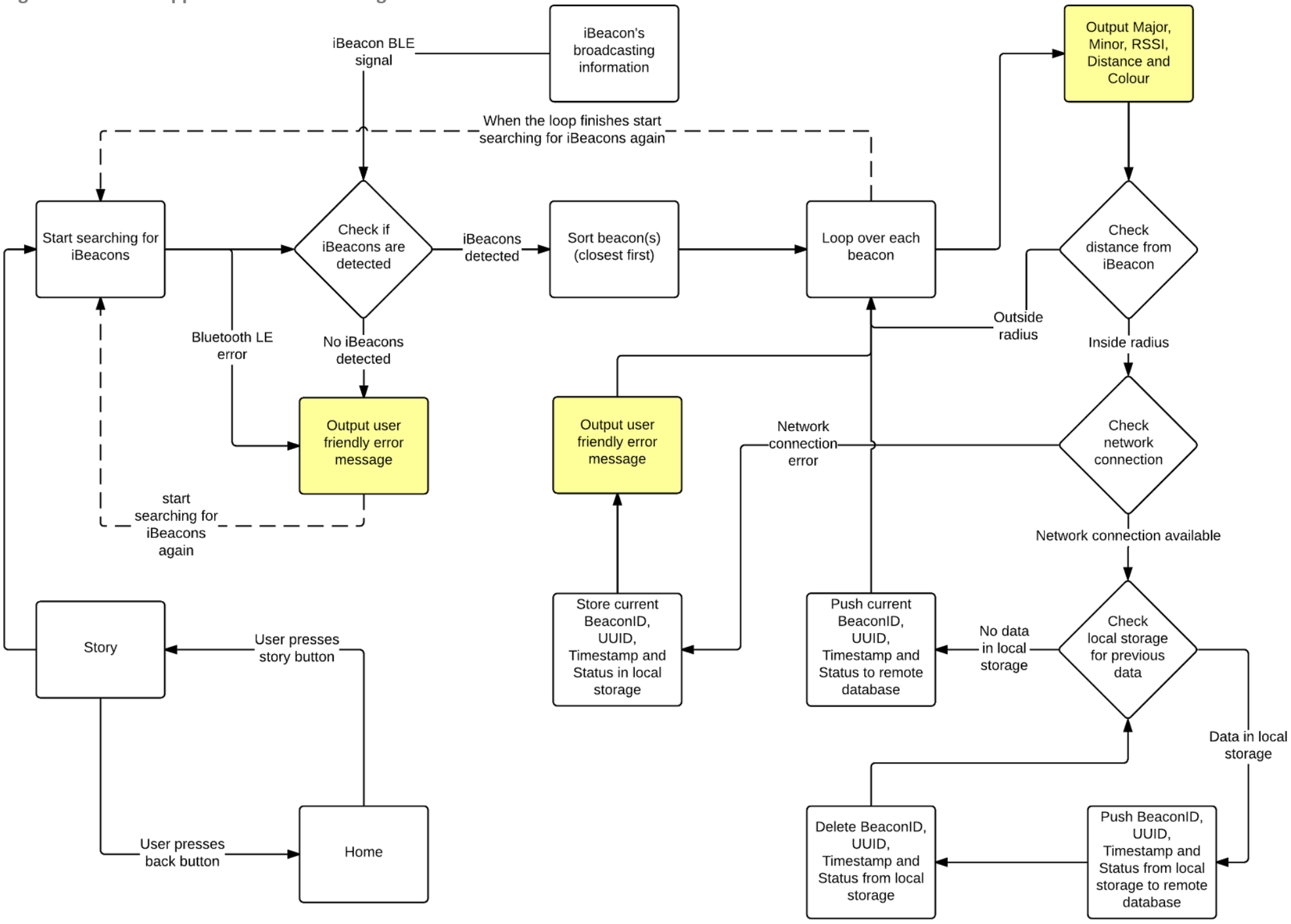
The main purpose of the mobile application is to interact with the iBeacons. It must be available on iOS and desirably Android. It will need to tell the user how far away they are from each of the iBeacons (content points). From the original investigation I discovered that the best way to deliver content was through radii and zones. I also found the optimum distance, advertising interval and position for the iBeacons. All of these findings will be used when developing this

prototype. When the user enters a specific radius of the iBeacons it will notify the user via a message on the application and send user behaviour data to a database.

10.1.1 Data flow diagram

The purpose of Figure 25 is to illustrate the flow of the mobile application. The square boxes represent a process. The diamond boxes represent a decision. The solid arrows show the flow of the system. The dashed arrows are used to highlight when the system returns back to the start. The highlighted yellow square boxes are output that the user can see.

Figure 25: Mobile application data flow diagram



The mobile application starts at the Home page. From there the user can navigate to the Story page. The user can navigate back from the story page to the home page.

Once on the story page the mobile application will start searching for iBeacons. If there is a problem with Bluetooth LE (BLE) on the smartphone a user-friendly error message will be output.

If there are no problems with BLE it will check to see if iBeacons are detected. If no iBeacon are detected a user-friendly error message will be output. To ensure this is carried out as quickly as possible so content is accurate and up to date I will only use parts of the Estimote API that are required.

If iBeacons are detected they will be sorted in ascending order based on their distance (closest first).

Once the iBeacons have been sorted they will be looped.

First the Major, Minor, RSSI, Distance and Colour will be output. Then the distance will be checked against the relevant iBeacon's radius.

If it is outside the radius it will return to the loop and look at the next iBeacon. If it is inside the iBeacon radius it will then check for a network connection.

If there is a network connection error it will store the BeaconID (concatenate major and minor), UUID, TimeStamp and Status in local storage and then output a user-friendly message. It will then return to the loop and look at the next iBeacon.

If a network connection is available it will check local storage for previous data that may have failed to send.

If there is data in local storage it will push BeaconID, UUID, TimeStamp and Status to the server. These four pieces of data are essential for the system management interface so it can uniquely identify the beacons and users and also know when they entered and exited different radii. Once it has been pushed to the server it will then be deleted from local storage. After this it will return to check local storage.

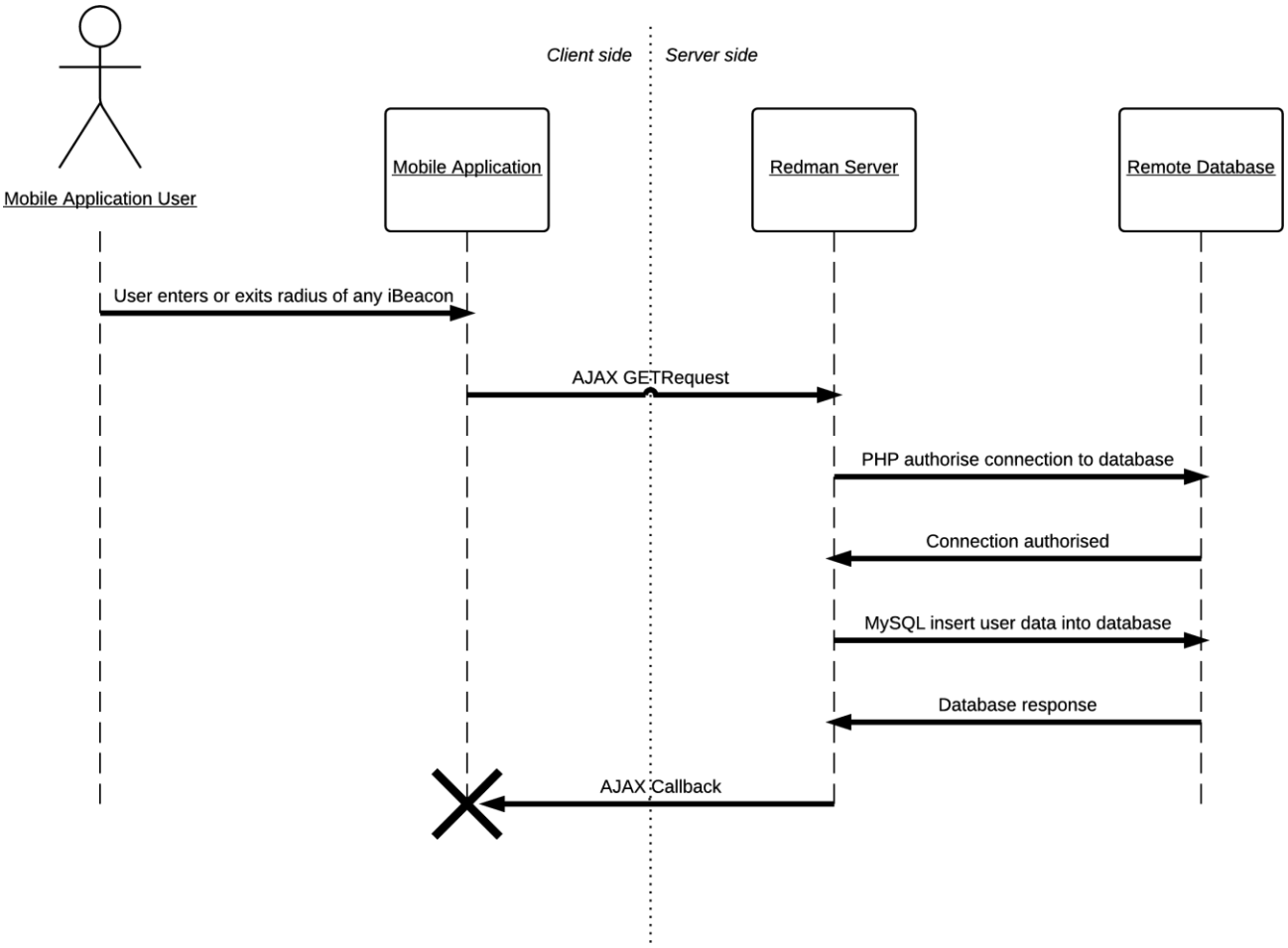
If there is no data in local storage the current BeaconID, UUID, TimeStamp and Status will be sent to the server. It will then return to the loop and look at the next iBeacon.

Once the loop has been finished the mobile application will return to the start and begin searching for iBeacons again.

10.1.2 Sequence diagram

The purpose of Figure 26 is to demonstrate how I plan to send user behaviour data from the mobile application to the remote database. The stickman represents an actor (the mobile application user). The rectangles and dashed lines represent objects and the "X" represents deletion (the end). The arrows with a description above them represent the different stages of the system. The system has a client side and server side which are separated by a dotted line and the sequence goes down the page with the first process in time at the top and the last process at the bottom.

Figure 26: Mobile application sequence diagram



Every time the user enters or exits the radius their information will need to be sent to a remote database so the Yellobrick employee can track them and deliver other types of content away from the application. To do this I plan to run an AJAX GET request from the mobile application to interact with a PHP file located on a server (called Redman). To send data from the mobile application to the PHP file I will need to use JSON with padding (JSONP). This is a communication technique used in JavaScript programs to request data from a server in a different domain, something prohibited by typical web browsers because of the same-origin policy. The PHP file will then attempt to establish a connection with the remote database using login credentials. Once a connection is authorised a MySQL insert query will be run to insert the user behaviour data into the database. After this has been completed a database response will be returned to the mobile application via an AJAX callback function that will notify the mobile application of any issues. If there are any issues the data will be stored in local storage.

10.1.3 Delivering content

To deliver content through the mobile application using the users physical location I will use both the radius and zone methods described in sections 7.2 and 7.3. These methods were both very reliable and only delivered content when the user entered a specific radius of an object. The radius method will be used to notify the user when they enter a specific iBeacon vicinity and the zone method will be used to deliver audio to the user at different volumes. The maximum radius and outer zone distance I will use for the iBeacons will be two metres as this was the maximum distance recommended from section 7.6.

Each of the iBeacons will have their advertising interval set to 750ms, as this is the most suitable from my findings in the section 7.4. Where it's possible the iBeacons will be placed at chest height, as proved in section 7.5.

For the system management interface content to work data will need to be pushed to the server. This will work by using the radius method which will then call a function to send user behaviour data to the server. The radius will be set at two metres for beacon 1,2,3 and set to one metre for beacon 4.

10.1.4 User interface design

One of the main requirements from the Yellobrick company was for the system to have a user-friendly interface. To ensure this I will make the mobile application simple, responsive, consistent, intuitive and forgiving. The prototypes from phase one will be used as a basis for these designs.

Home page

The purpose of the home page is to provide navigation to the story page. The reason I didn't load up the story page straight away was so the user had a bit of freedom and could choose when they wanted to start the interactive experience.

Figure 27: Mobile application home page design

The "Beacon Alert" section is the title throughout the application. The blue "Story" section is a button that will navigate the user to the story page and start the experience. The design is clean simple and intuitive with just two sections clearly separated by vibrant colours. This page is responsive as the title and button sizes change depending on the font size. To keep the application consistent I will use the same colour scheme, layout and font throughout the application. The user can press back on the story page which makes the system forgiving. See Figure 27.

Story page

The purpose of this page is to start the interactive experience of the user and communicate with the iBeacons. In the back-end it will determine the position of the user and send data to the server. It will deliver iBeacon information and content to the user in the front-end.

Figure 28: Mobile application story page design

This page has a very similar layout to the home page, but instead of a "Story" button it has a back button allowing the user to stop the interactive experience and return to the home page. The content section will display information about all the iBeacons in proximity including their major, minor, distance, RSSI and colour. A message will be delivered to the user If they enter or exit a radius or zone. If the interaction fails because of a problem with the internet, BLE signal or if no iBeacons can be detected a user friend message will be output notifying the user. See

Figure 28.

10.2 System management interface

The system management interface needs to allow Yellobrick employees to track user behaviour meaning they could view the full enter and exit history of the system using data from the database. They should also be able to set up screens which would deliver content to users away from the mobile application. This will include a horror page which will confuse the user in a dark horror scene type setting. A monitor section will also be included which will play different videos when users enter the radius of specific iBeacons. Another section will consider the behaviour of unique users and will only trigger when iBeacons are visited in a specific order. The system management interface system is primarily for the Yellobrick employee so all error messages are directed to the employee. The only information the mobile application users should be interested in is viewing the content when the employee loads the system up onto screens.

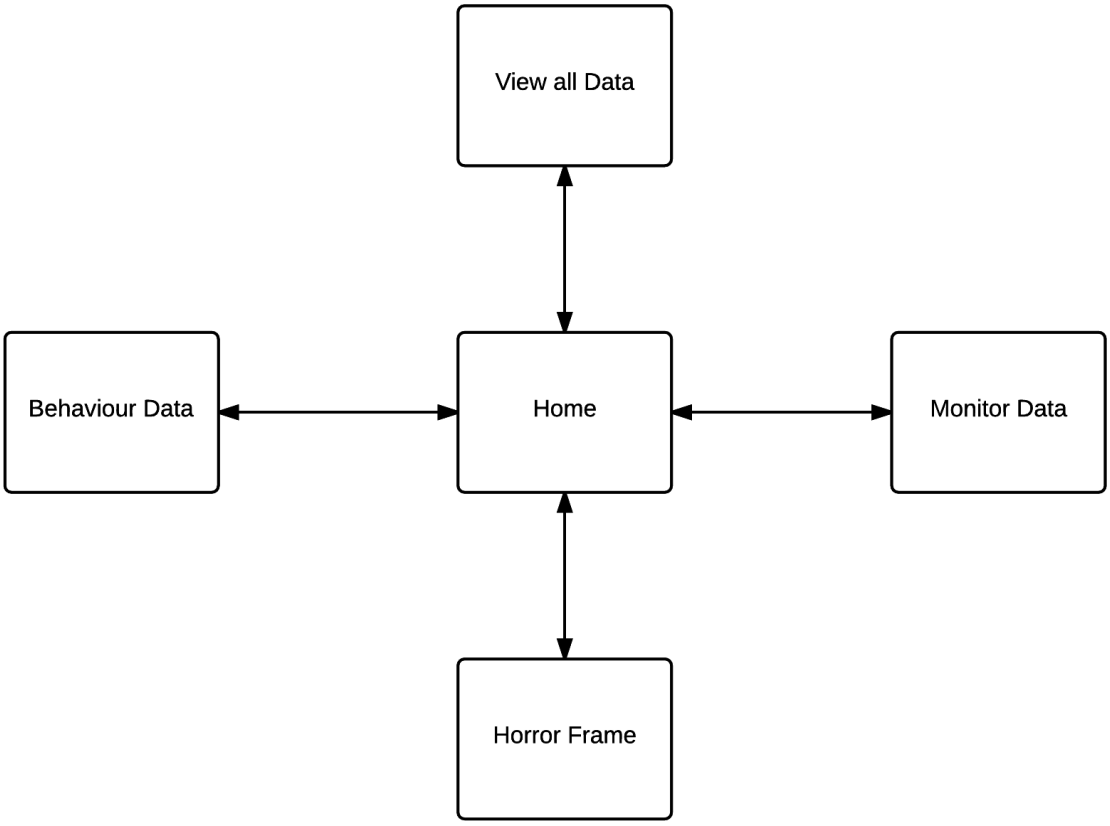
10.2.1 Dataflow diagrams

Like with the mobile application diagram previously these diagrams will demonstrate the flow of system management interface using the same objects as described in section 10.1.1.

Home

From the home page the Yellobrick employee will be able to View all data, monitor data and view the horror frame story. The employee will be able to navigate back to the home page by pressing the back button. Figure 29 illustrates the home page data flow.

Figure 29: System management interface home page data flow

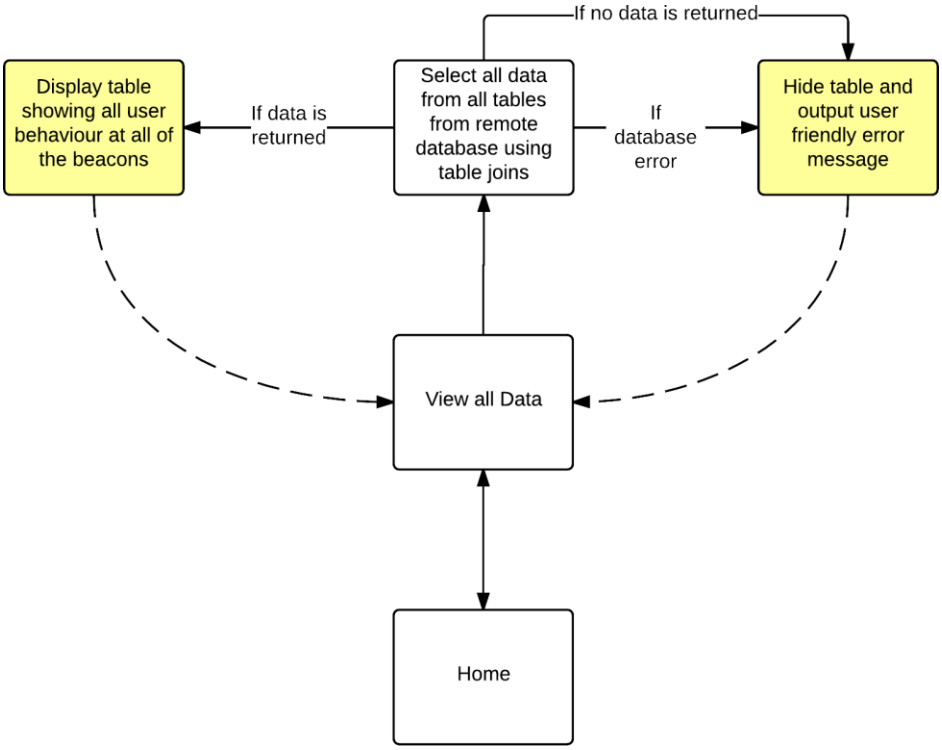


The view all data, monitor data, horror frame and behaviour data sections are complex and will deliver different content to the employee and user. These four sections will be explained in detail later in this section.

View all data

The purpose of the view all data section is for Yellowbrick employees to view different users dwell time. It will show the complete behaviour history of every user. This page will be set up by the Yellobrick employee and then displayed on a monitor. Mobile application users will then be able to view this to enhance their experience. Figure 30 illustrates the view all data flow.

Figure 30: System management interface view all data flow



Once on the view all data page the system management interface will query the server and select all the details for every entry and exit record in the database. This will include the LogID, BeaconID, UUID, Timestamp, Status, RoomID, RoomName, LocationID, LocationName, Longitude and Latitude. This ensures efficiency as all of these fields are needed to link together the tables and display the BeaconID, UUID, Date, Time, Status, Room Name, Location Name, Latitude and Longitude to the Yellobrick employee so they can track users.

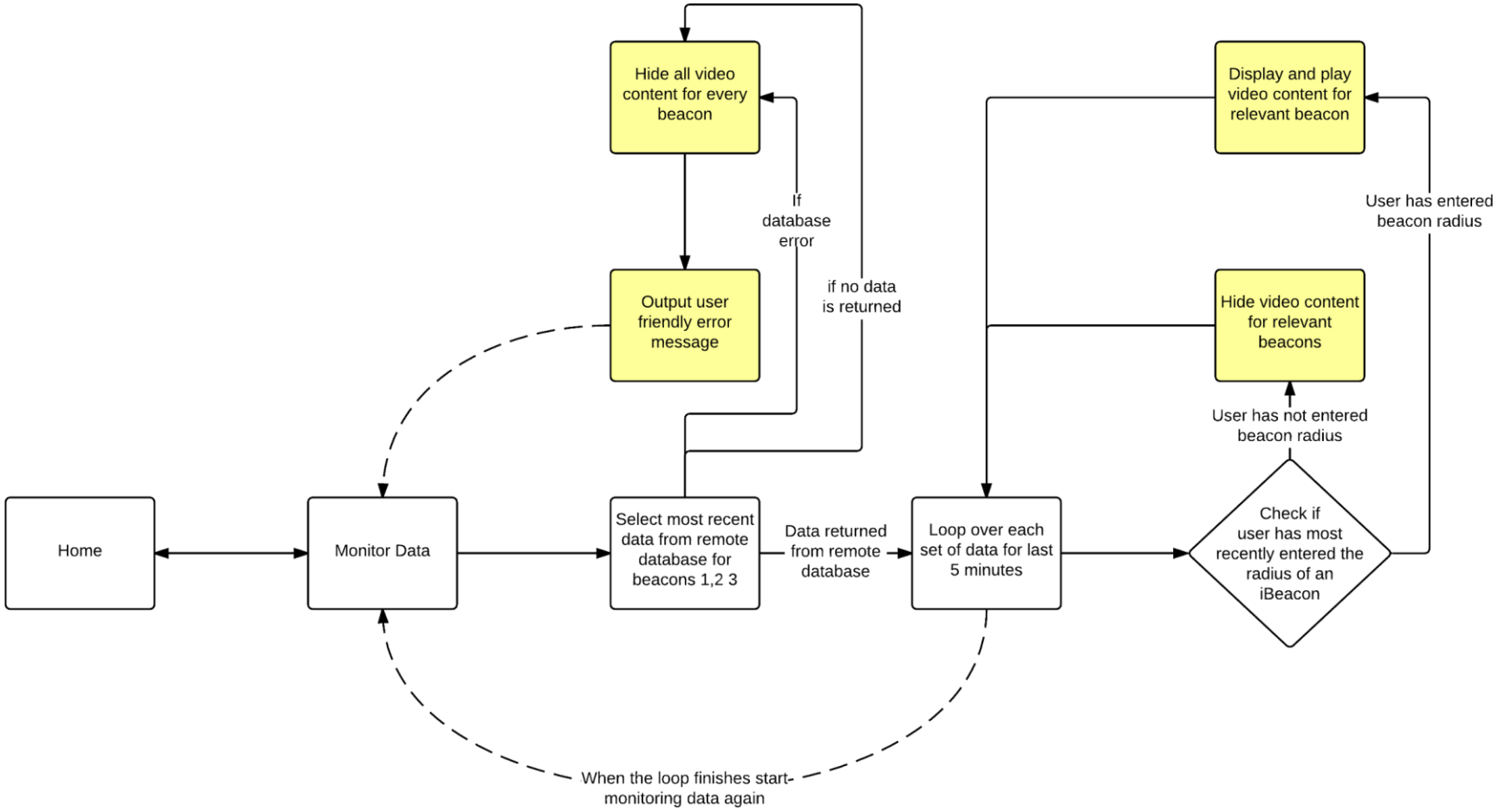
If there is a database error or no data is returned the table will be hidden and a user-friendly message will be output. If data is returned it will be displayed in a table.

The user will have an option to manually refresh the page which will repeat the above.

Monitor data

The purpose of this page is to deliver content to users using live data from the database. If a user enters the radius of an iBeacon a video will play to give the user a more interactive experience. This page will be set up by the Yellobrick employee so it's displayed on a monitor. Mobile application users will then be able to view this to enhance their experience. Figure 31 illustrates the monitor data flow.

Figure 31: System management interface monitor data flow



On the monitor data page the system management interface will query the data from the live database for beacons 1,2 and 3. The data needed will include the LogID, BeaconID, UUID, Timestamp and Status. This ensures efficiency as all of this data will be used when deciding whether to deliver content or not so there is no waste data.

If there is a database error or no data is returned it will hide all video content for every beacon and output a user-friendly error message. It will then start to monitor data again.

If data is returned it will loop over each set of data for the last five minutes.

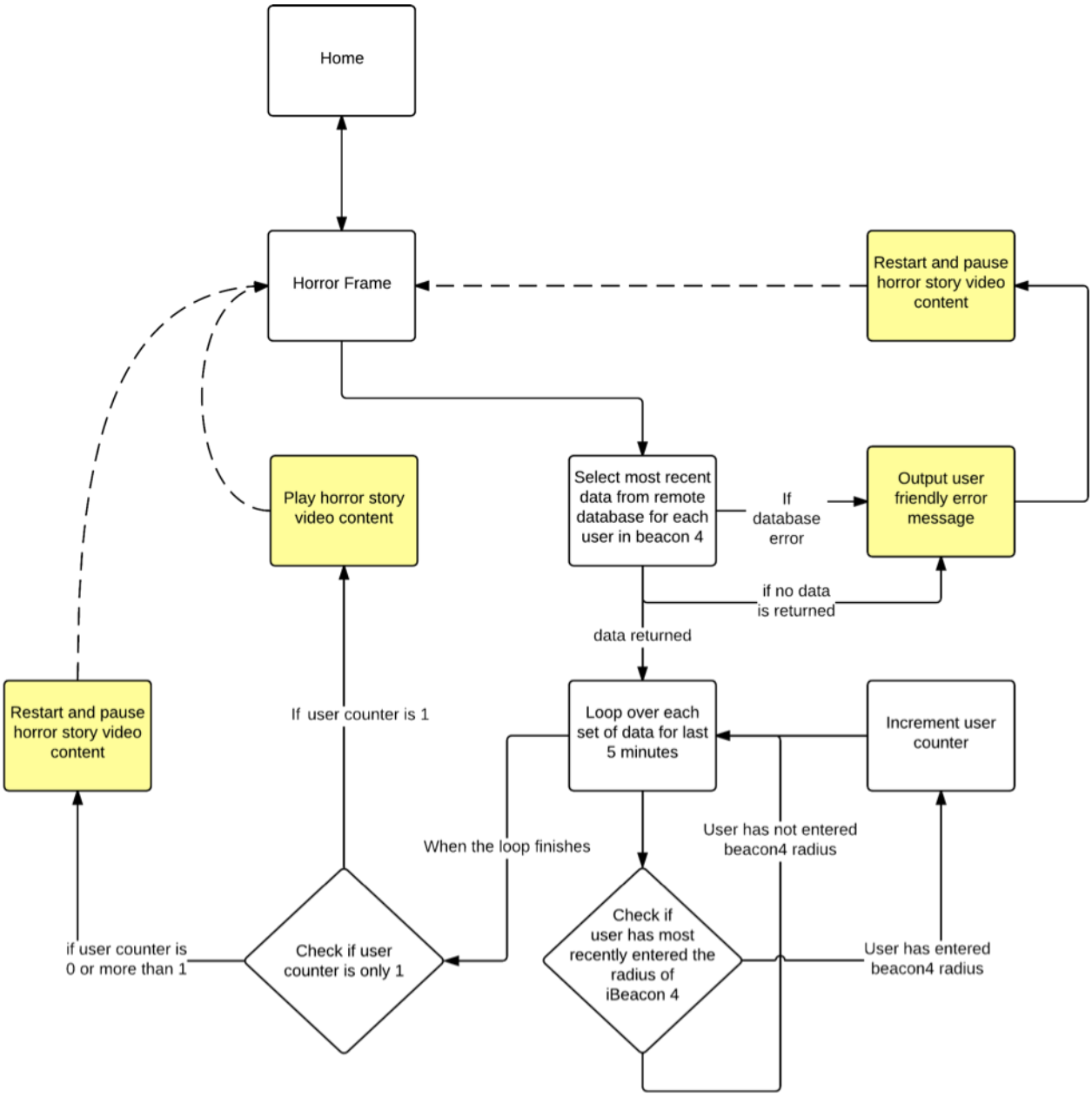
If a user's last action was to enter the radius of an iBeacon it will display and play the video for the relevant beacon. If no user has entered the radius of an iBeacon the relevant video will be hidden.

Once the loop has been finished the mobile application will return to the start and begin monitoring data again.

Horror frame page

The purpose of this horror page is to confuse the user in a dark horror scene type. This page will be set up by the Yellobrick employee and displayed on a monitor. Mobile application users will then be able to view this to enhance their experience. Figure 32 illustrates the horror frame data flow.

Figure 32: System management interface horror frame data flow



Once on the horror frame page the system will query the database to select the most recent user activity data when interacting with beacon 4. Similarly to the monitor data page the LogID, BeaconID, UUID, Timestamp and Status will be required. Again this ensures efficiency as all of this data will be used when deciding whether to deliver content or not to avoid waste data.

If there is a database error or no data is returned a user-friendly message with be output and the horror story video content will be restarted and pause to look like a picture.

If data is returned it will loop over each set of data for the last 5 minutes.

If a user's last action was to enter the radius of iBeacon 4 it will increment the user counter. It will then return back to the loop and look at the next set of data.

If a user's last action was to exit the radius of iBeacon 4 it does nothing and returns back to the loop to look at the next set of data.

When the loop finishes it will check the user count is equal to one.

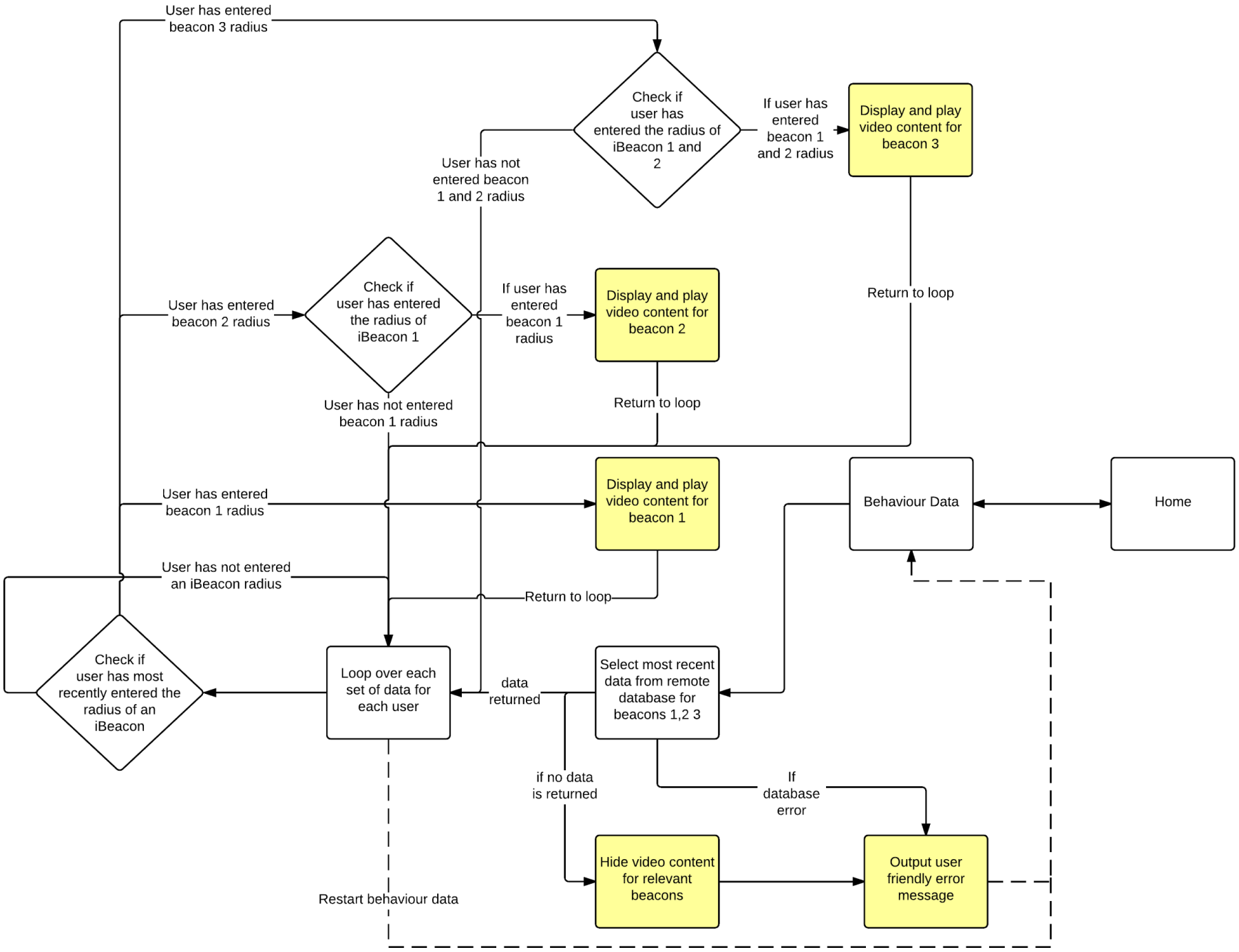
If it is 0 or greater than one the horror story video content will be restarted and paused to look like a picture. It will then start the horror frame process all over again.

If it is equal to one it will continuously play the horror video on repeat. It will then start the horror frame process all over again.

Behaviour data

The purpose of this page is to only display content when a user visits the iBeacons in specific order. For example they need to visit beacon 1 before they can view beacon 2 content and they need to visit beacon 1 and 2 before they can view beacon 3 content. This page will be set up by the Yellobrick employee so it's displayed on a monitor. Mobile application users will then be able to view this to enhance their experience. Figure 33 illustrates the monitor data flow.

Figure 33: System management interface behaviour data flow



On the behaviour data page the system will query the database for the most recent user activity data for beacons 1,2 and 3. Like the monitor data page and horror frame page the LogID, BeaconID, UUID, Timestamp and Status will be required. As with the other pages this ensures efficiency as all of this data will be used when deciding whether to deliver content or not to avoid waste data.

If there is a database error or no data is returned it will hide all video content for every beacon and output a user-friendly error message. It will then start the behaviour data process again.

If data is returned it will loop over each set of data checking to see if the user's last action was entering the radius of any of the three iBeacons.

If the user has entered the iBeacon 1 radius it will display and play the video content for iBeacon 1.

If the user has entered the iBeacon 2 radius it will check to see if iBeacon 1 has been previously visited through cookies which are unique for each user. If it has it will play the video content for iBeacon 2, but if not it will return to the loop.

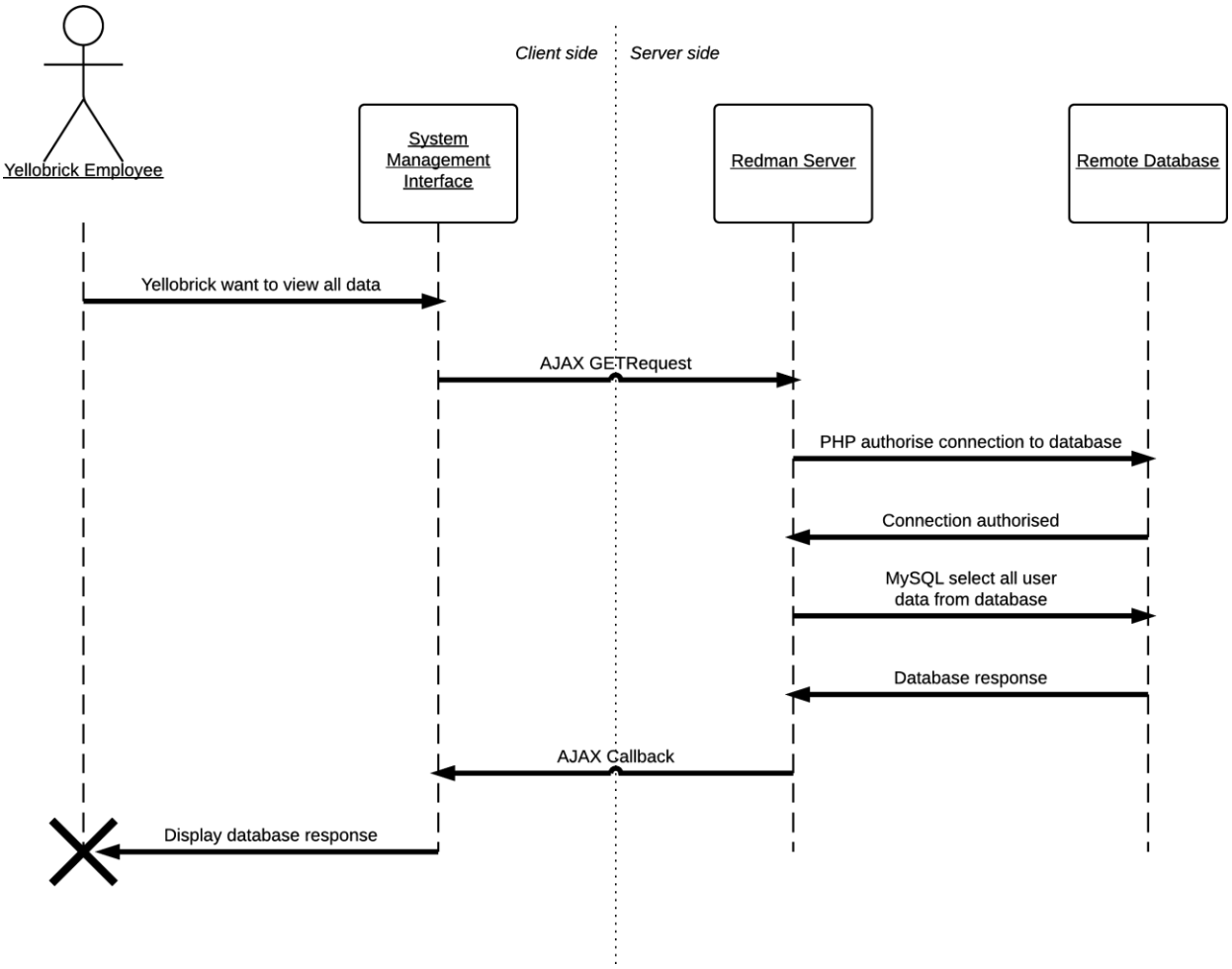
If the user has entered the iBeacon 3 radius it will check to see if iBeacon 1 and 2 have been previously visited by again looking at the cookies. If they have it will play the video content for iBeacon 3, but if not it will return to the loop.

Once the loop has been finished the mobile application will return to the start and begin the behaviour process again.

10.2.2 Sequence diagram

Figure 34 demonstrates how I plan to retrieve data from the system management interface. As with the mobile application sequence diagram in section 10.1.2 the symbols have the same meaning.

Figure 34: System management interface sequence diagram



User behaviour data will be requested from the system management interface for the view all data and three content pages. The sequence will be similar for all four, but with different queries. This diagram demonstrates how the data will be grabbed using the view all data page. As you can see the diagram is quite similar to the mobile application sequence diagram mentioned previously as I already have a connection established I just change the query and response. I will run an AJAX GET request from the system management interface to interact with a PHP file located on the Redman server. JSONP will again be used to transfer the data because of cross-domain restrictions. The PHP file will then attempt to establish a connection with the remote database using login credentials. Once a connection is authorised a MySQL select query will be run to select user behaviour data from the database. After this has been completed a database response will be returned to the system management interface via an AJAX callback function containing all the data from the database. If there are any issues this error message will be parsed through the AJAX callback function as the response and notify the Yellobrick employee.

10.2.3 Delivering content

To deliver content through the system management interface using the user's physical location I will use the user's enter and exit data available from the database. This data will have been pushed to the server only when the user enters and exits the radius specified in the mobile application.

This means the system management interface will only need to check the last five minutes of data to determine whether to deliver content to the users as the radii check will have already been done in the mobile application.

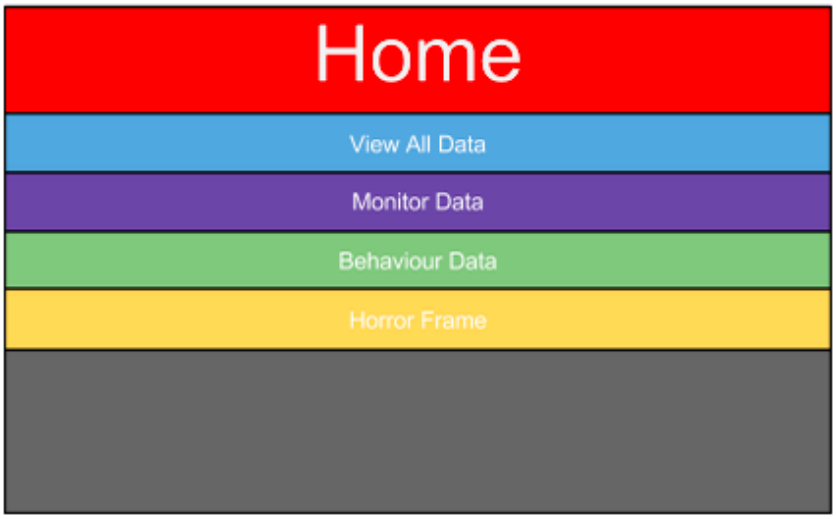
10.2.4 User interface design

Like with the mobile application the system management interface needs to be simple, responsive, consistent, intuitive and forgiving. The system management interface also needs to be portable so it works across all major browsers. This means it will need to work on Chrome, Internet Explorer, Safari and Firefox.

Home page:

The purpose of this page is to provide navigation to the different sections of the system management interface.

Figure 35: System management interface home page design



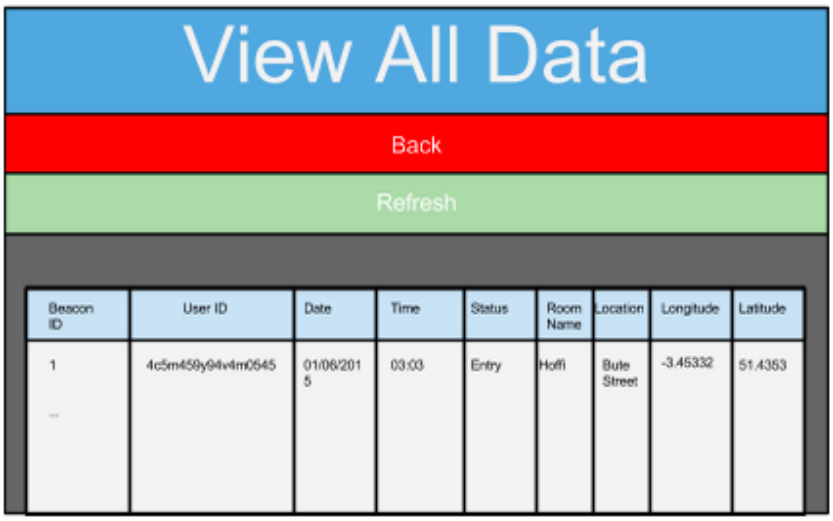
The big "Home" section at the top notifies the user where they are. Each of the sub sections below are buttons to the different pages within the system. The design is clean simple and intuitive with each section being clearly separated using vibrant colours. The layout is responsive as the content section size changes depending on font size. The colour scheme, layout and font is the same through the system

management interface to keep it consistent. The user has the ability to go back from each of the destination pages which makes the system forgiving. See Figure 35.

View all data page

This page must show the entire system history by selecting all the data from the database. This includes the enter and exit times at different locations which allows Yellobrick employees to track users. This should be used by the Yellobrick employees for their own use only and not shown to normal users.

Figure 36: System management interface view all data page design

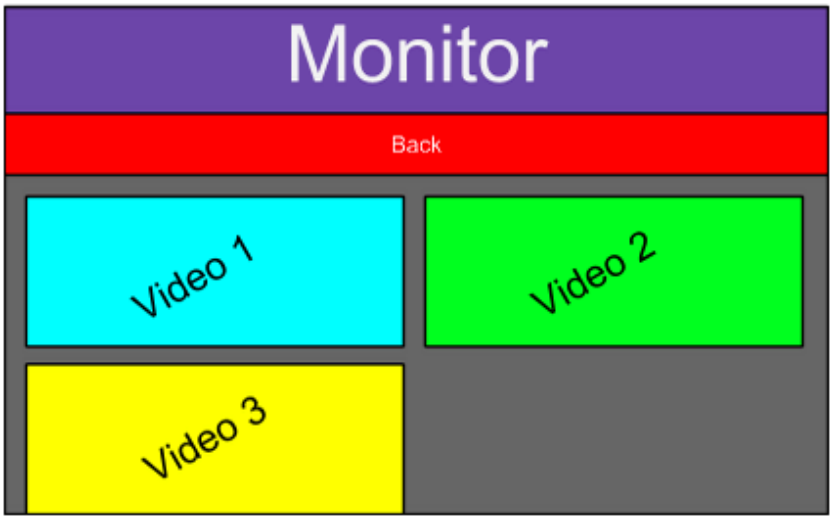


Each of these subpages has a similar layout with the page title at the top and a back button below which takes the user back to the home page. In the back-end a select query will grab data from the database and this data will then be displayed in the front end via a table so it is easy for the Yellobrick employee to understand. The refresh button on this page updates the table using the latest data from the database. The design is clean, simple, intuitive and responsive like the other pages. If there is a problem and data cannot be retrieved from the database or the database is empty a user-friendly error message will be displayed via the interface notifying the user of the problem making this page forgiving. See Figure 36.

Monitor page

This page must deliver content to users using live data from the database. If a user enters the radius of an iBeacon a video will play to give the user a more interactive experience. This page allows the Yellobrick employee to specify the content that will be delivered to the mobile user. Mobile application users will then be able to view this to enhance their experience.

Figure 37: System management interface monitor page design

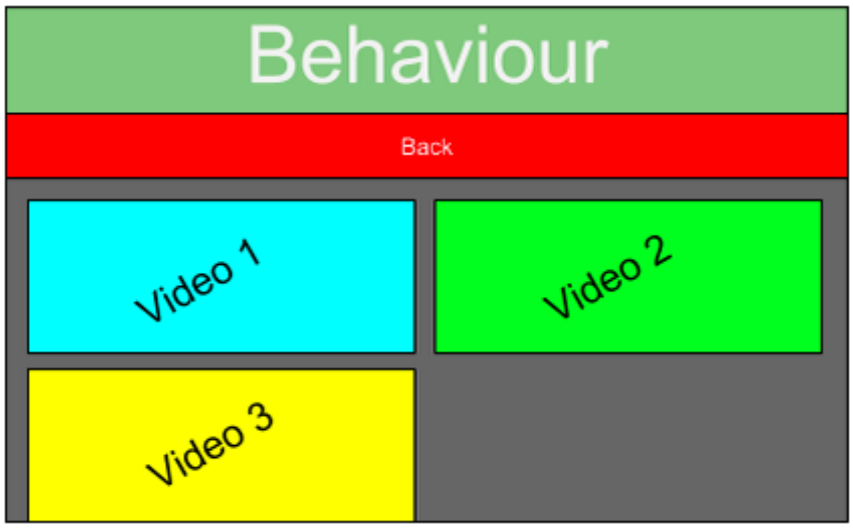


Like the other subsection pages it has a title, back button and is clean, simple, intuitive and responsive. It will work by querying the database every few seconds checking to see if any users are within the radius of any of the three iBeacons. If a user's last action is entering the radius of an iBeacon the relevant video will be played. If a user's last action was to exit the radius of an iBeacon the relevant video will be hidden. Like the other pages if there is a problem with the database the user will be notified. If there are no users inside any of the radii a user-friendly message will be output via the interface saying there is no user inside any iBeacon radius. See Figure 37.

Behaviour page

This page must display content when a user visits the iBeacons in specific order. For example they need to visit beacon 1 before they can view beacon 2 content and they need to visit beacon 1 and 2 before they can view beacon 3 content. This page will be set up by the Yellobrick employee so it's displayed on a monitor. Mobile application users will then be able to view this to enhance their experience.

Figure 38: System management interface behaviour page design

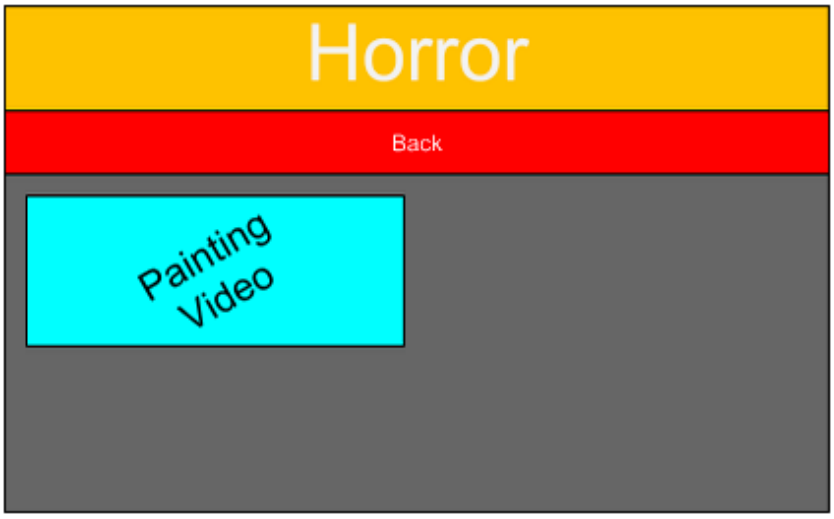


Like the monitor page it has a title, back button and is clean, simple, intuitive, responsive and it will query the database. This though will also check to see if the user has entered relevant iBeacons before delivering content. If the user hasn't accessed the previous iBeacons they will be notified through a user-friendly message. As with the other pages if there is a problem with the database or there are no users inside any radius the user will be notified via the interface. See Figure 38.

Horror page

The horror page needs to confuse the user in a dark horror scene type. This page will be set up by the Yellobrick employee so it's displayed on a monitor. Mobile application users will then be able to view this to enhance their experience.

Figure 39: System management interface horror page design



Like the monitor page it has a title, back button and is clean, simple, intuitive and responsive. A video will be displayed at all times and set to pause. Then when one user enters the radius of one of the iBeacons it will play the video. If multiple or zero users are inside the iBeacon radius the video will pause

and return to the start. This will query the database like the monitor page, but only return the most recent row for each user and their interaction with iBeacon 4. Then it will count how many users are currently inside the radius of iBeacon 4. As previously if there is a problem with the database the user will be notified via the interface. See Figure 39.

10.3 Database system

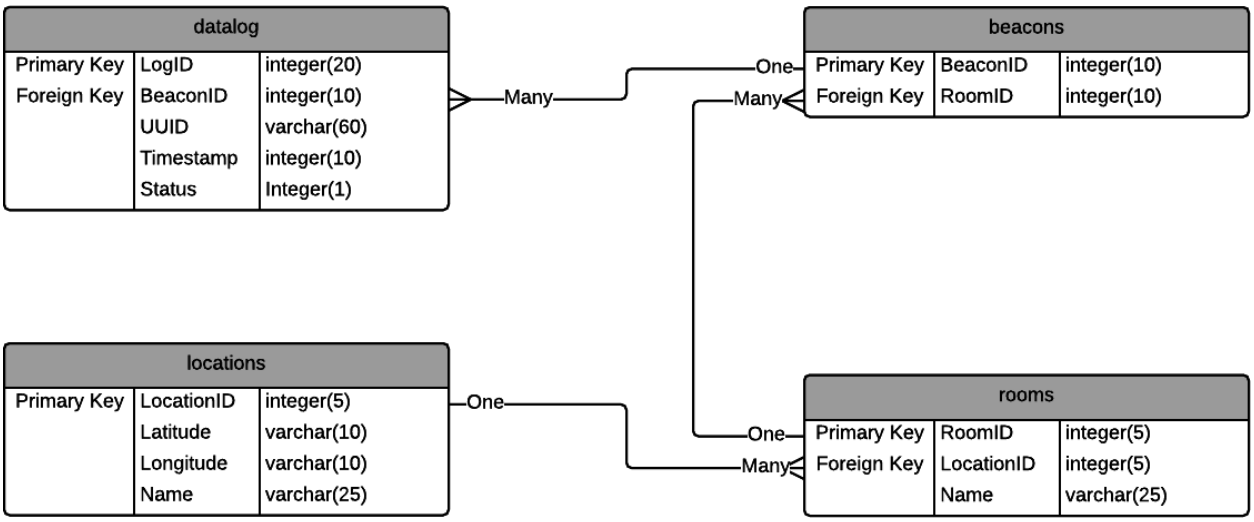
The database system must store the data sent from the mobile application. The system management interface will then query this database to view user behaviour and deliver content. This is one of the main requirements and is essential so the system management interface works.

I used normalisation techniques when designing my database to ensure data stored is accurate, efficient to access and to avoid data redundancy. I designed the database using smaller tables rather than one large table. I have used foreign keys (blue rows) to reference primary keys (yellow rows) of other tables. See Figure 40.

10.3.1 Entity Relationship diagram

The entity relationship diagram describes the data and shows the relationships between the different entities (tables) in the database.

Figure 40: Data system entity relationship diagram

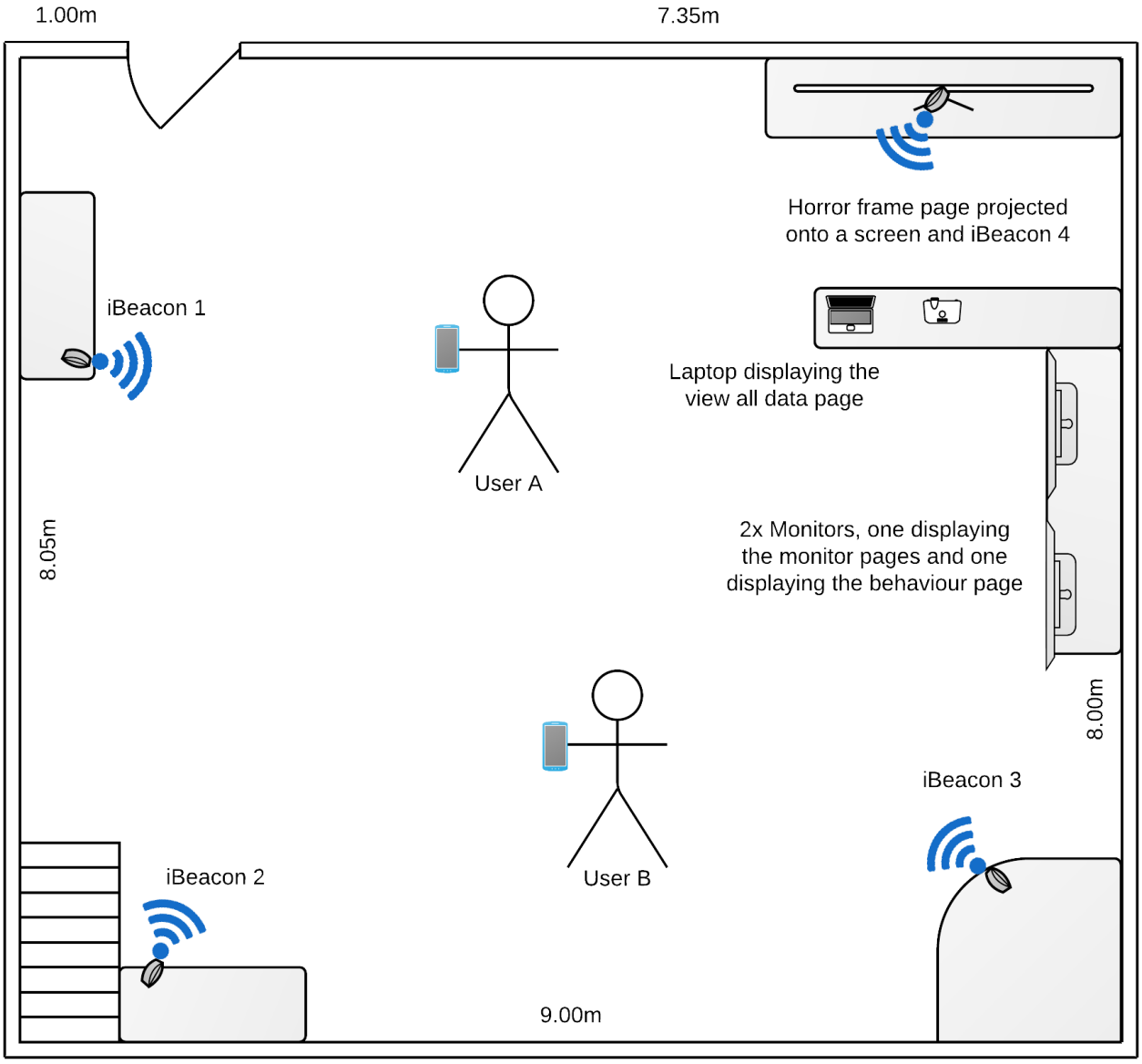


The first row in each entity describes the name of the table. Each row thereafter describes a field in that table. The first column defines the state of the field, explaining whether it is a primary key, foreign key or normal field. The second column describes the field name. The third column described the data type and in brackets describes the length of each field. The arrows between the table show the one to many relationships between the tables.

10.4 Environment

Here I will be using my findings from phase one to produce a recommended environment for the final prototype. See Figure 41. There will be two mobile application users both using iOS with access to Wi-Fi and Bluetooth. The recommended environment will include four iBeacons, four monitors with access to the system management interface and a network connection. The room dimensions are 8.05m x 9.00m. The shaded areas represent tables and desks. The small oval object with a blue signal represents the iBeacons. The actor holding a phone represents the users. The other electrical devices including laptops, monitors, projectors and projector screens are all located on tables.

Figure 41: Recommended environment design



Each iBeacon will be placed at least 5 metres apart and at chest height (approximately 1.5 metres). With iBeacons 1,2 and 3 all delivering content at 2 metres it is important to leave a bit of leeway. In the example environment above they are approximately 6-8 metres apart.

The monitors displaying the behaviour and monitor pages will need to be viewable from all areas in the room so when users interact with iBeacons 1,2 and 3 they can see the content.

The Yellobrick employee will be able to view all the user behaviour data from the laptop screen.

The horror frame screen will be displayed on a projector from the laptop and set to full screen to look like a painting. IBeacon 4 will be separate to the other 3 iBeacons and placed in a corner in a dark setting.

The two users will have the freedom to walk around the room and interact with the different iBeacons and experience the different content.

11 Implementation Phase Two: Yellobrick Prototype

Here I will discuss how I developed the Yellobrick prototype. This will include a section on justifying the application development tool I decided to use. I will discuss database system structure and how I created it. I will then elaborate on how I implemented the mobile application and the system management interface where I will highlight the complex and important pieces of code.

11.1 Application development tools

I will develop the mobile application using hybrid development. This means developers can build the majority of the project using HTML5, CSS and JavaScript, but still be able to call into native code when necessary. The other option is to develop an app using the native language of the platform. IOS uses Objective-C or Swift and Android uses Java. The advantages are described in Table 5.

Table 5: Native vs Hybrid application development (Ziflaj, 2014)

Reasons for Native	Reasons for Hybrid
Better performance. It is compiled into machine code which results in the best performance you can get from a smartphone.	Faster to develop.
More up to date API's.	Simpler to develop.
Full access to the phones Hardware.	Easier to maintain.
Doesn't depend on the native browser.	Much fast to develop cross platform applications.

Hybrid development is more suitable to this project as I don't have much time and one of the desirable requirements of the project is for it to work on both Android and iOS. There are two main competitors in the hybrid development field, Cordova and Appcelerator Titanium. I will be using Cordova as I have used this previously so won't need to spend time understanding and learning the technology.

Cordova is a platform for building native mobile applications using HTML, CSS and JavaScript. It is a set of device APIs that allow a mobile app developer to access native functions such as the camera or accelerometer from JavaScript. Cordova is effectively a web view that occupies the complete screen and runs in the native container. It uses the same web view that is used by native operating systems which means that only the native containers change according to the OS and internally the web pages remain the same. This means the Cordova libraries can communicate with the native framework of the respective operating system. Applications using Cordova are still packaged as apps using the platform SDK's so can be made available on the platforms app store like a native app.

In order to access hardware and software information from the smartphone I will need to use Cordova plugins (Cordova, 2014). During the development stage I will use the console plugin to help with debugging. One of the requirements is to uniquely identify users, which will be done using the device's UUID. To get the UUID I will need access to the device settings which means I will need the Device plugin. To access read and write to the device I will need the file plugin installed. This will allow me to save data in local storage. One of the requirements is to deliver content to the user via the mobile application. To do this I will need to play audio devices meaning the Cordova Media plugin is required. The client wants the mobile application to be forgiving and alert users when there are errors. Part of the mobile application functionality includes sending data to a web server over the internet. If there is no internet this will not work and crash the application. To stop this happening I will run a check which will notify the user when the network connection isn't available. To do this I need access to the smartphones network information, this can be accessed using the Network Information plugin. To interact with the Estimote iBeacons I will need the EstimoteBeacons API which is available as a Cordova Plugin.

I used FileZilla FTP client to manage files on the remote server as I have used this previously meaning I didn't have to spend time learning how to use a new FTP client.

I will be coding the majority of the system using the text editor Sublime Text. The editor is clean, functional, fast and can run on all the main operating systems. It also supports plugins snippets and many other things that will help me when coding the system.

During the testing stages on iOS I will use Xcode as this comes with many simulators for lots of different iPhone models. Xcode also makes it much easier when loading the application onto

the mobile device. Hoffi have given me access to their Apple developers account which allows me to run the mobile application on real devices.

To backup my project I will be copying it over to the Hoffi server multiple times a day. This server is then backed up by Hoffi each night and stored on a remote server. This will result in multiple copies in case something goes wrong. I looked into other backup methods including Git, but seeing as this was a solo project and I haven't had much experience with GitHub I decided to use Hoffi's server instead.

For my database I decided to use PHPMyAdmin as it is a free tool that handles the administration of MySQL in a web browser. It was already installed on the Hoffi server which meant I didn't have to waste time setting up a new database system.

To interact with the database I will use a combination AJAX, JSON, JSONP, PHP and MySQL. To send data from the mobile app to the database JSONP will need to be used to allow for cross-site AJAX with JSON data. Normal JSON will not work when sending data from a mobile device to a remote server so JSON with padding is required (JSONP). This will send data to a remote PHP file located on a server. This PHP will then insert data into the database using a MySQL insert query. To retrieve data from the database so it can be viewed on the system management interface a similar AJAX call with JSONP will be run. The remote PHP file will then use a MySQL select query to select the relevant data from the database.

The horror page section of the system management interface requires some sort of local storage so unique users can be tracked during sessions. This is required to check users have been to previous iBeacons delivering content. I will use JavaScript cookies to do this as the cookie only needs to be 39 bytes. Cookies are primarily for reading server-side data so they are more suitable to this.

I will use the JQuery JavaScript library for both the mobile application and system management interface as it promotes simplicity, loads faster and integrates well with cookies and local storage.

In the event a network connection is not available the data will need to be stored locally on the smartphone so it is not lost. To do this I will save it using local storage (HTML5 web storage). I have decided to use this over cookies as it is more secure and doesn't impact performance. The

data being stored in local storage will be BeaconID, UUID, Timestamp and Status. Once a network connection is available all data from local storage will be sent to the database on the server and then deleted from local storage.

One of the requirements is each user must be uniquely identified anonymously to avoid privacy issues. To do this the user's device Universal Unique Identifier (UUID) will need to be made anonymous. This could be done using a hashing algorithm which will still allow users to be uniquely identified, but it won't show the users actual UUID. A hashing algorithm maps digital data of any size to digital data of fixed size which allows that piece data uniquely identified
For example it will convert a UUID such as "c669310a-e113-11e4-8a00-1681e6b88ec1" to an MD5 hash such as "d41d8cd98f00b204b9800958ecf8427e"The same UUID will be converted to the same hash value every time allowing me to still uniquely identify users.

11.2 Database system

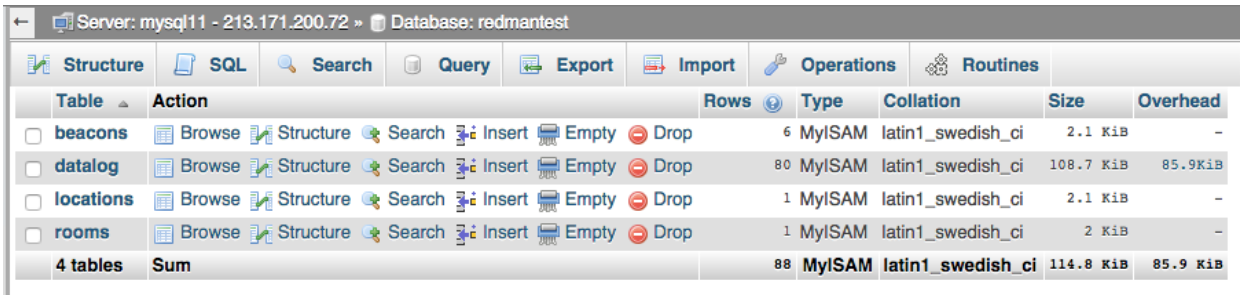
The full database structure can be found in the Appendix 4.

I used the same structure I designed in section 10.3.1. This eliminated redundant data and ensured data dependency resulting in efficiently organised data inside the database.

I created the redmantest database on Hoffi's database server using PHPMyAdmin. I then created the tables using the PHPMyAdmin interface, naming them locations, rooms, beacons and datalog. Figure 42 shows an overview of the database in PHPMyAdmin.

11.2.1 Structure overview

Figure 42: Database structure overview



11.2.2 Locations table

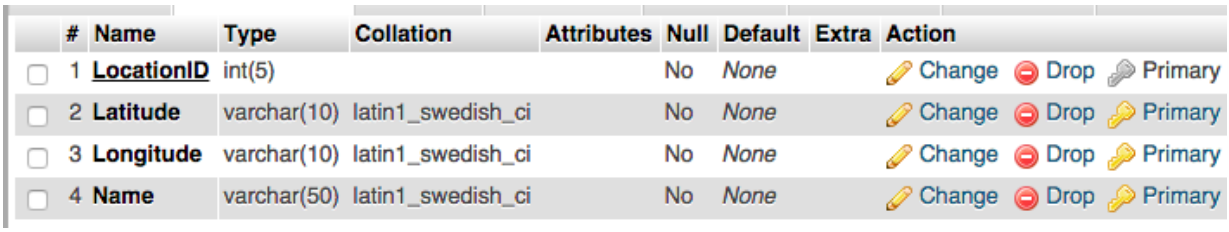
For the locations table I created four columns LocationID, Latitude, Longitude and Name. See Figure 43.

I set the LocationID to be the primary key as this is used to uniquely identify each location. The data type is integer with a length of 5 as this supports up to 99,999 locations.

The Latitude and Longitude columns will be VarChar (Variable Character field) data type at a length of 10 as they are a maximum of 10 characters long on Google when the sign, integers and decimal point are considered.

The Name column will be a long VarChar data type of length 50 to accommodate place names in South Wales.

Figure 43: Database locations table structure



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	LocationID	int(5)			No	None		Change Drop Primary
2	Latitude	varchar(10)	latin1_swedish_ci		No	None		Change Drop Primary
3	Longitude	varchar(10)	latin1_swedish_ci		No	None		Change Drop Primary
4	Name	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary

11.2.3 Rooms table

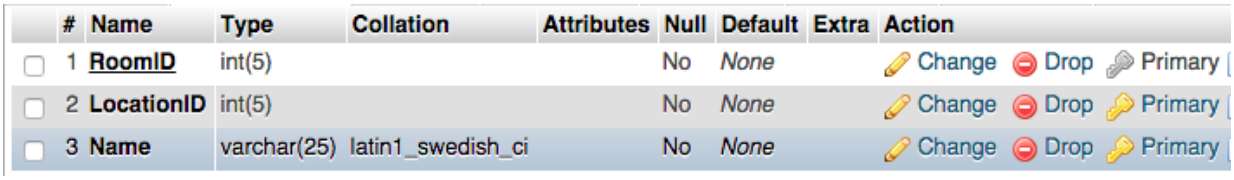
I created three columns for the rooms table, RoomID, LocationID and Name. See Figure 44.

I set the RoomID to be the primary key as this is used to uniquely identify each room. The data type is integer with a length of 5 as this supports up to 99,999 rooms which should be sufficient for the system.

The LocationID will be a foreign key linking this rooms table to the locations table. The data type and length match the LocationID from the locations table.

The Name column will be a long VarChar data type of length 25 which should be long enough to accommodate most room names.

Figure 44: Database rooms table structure



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	RoomID	int(5)			No	None		Change Drop Primary
2	LocationID	int(5)			No	None		Change Drop Primary
3	Name	varchar(25)	latin1_swedish_ci		No	None		Change Drop Primary

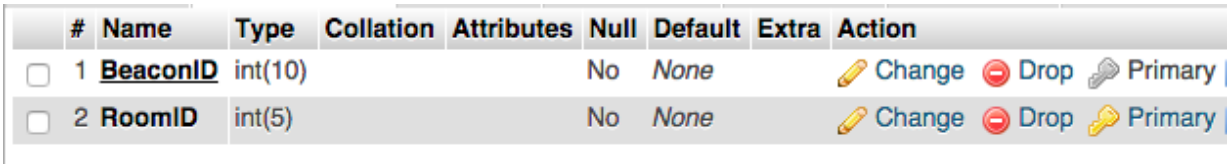
11.2.4 Beacons table

I created two columns for the beacons table, BeaconID and RoomID. See Figure 45.

The BeaconID is the primary key used to uniquely identify each iBeacon. The data type is integer with a length of 10 as the major and minor concatenated can be a maximum length of 10.

The RoomID will be a foreign key linking this beacons table to the rooms table. The data type and length match the RoomID from the rooms table.

Figure 45: Database beacons table structure



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	BeaconID	int(10)			No	None		Change Drop Primary
2	RoomID	int(5)			No	None		Change Drop Primary

11.2.5 Datalog table

I created five columns for the datalog table, LogID, BeaconID, UUID, Timestamp and Status. See Figure 46.

The LogID is the primary key used to uniquely identify each log record. The data type is integer with a length of 20 as there will be a lot of data logged in this table and a length of 20 will allow billions of unique LogID's.

The BeaconID will be a foreign key linking this datalog table to the beacons table. The data type and length match the BeaconID from the Rooms table.

The UUID column will be a VarChar field of length 60 to support the hashed UUID's.

The Timestamp will be an integer value of length 10 as the maximum timestamp length is 10 integers.

The Status will be a single integer value as only 1 or 0 will be stored. This saved space and is more efficient than using “Enter” or “Exit”

Figure 46: Database datalog table structure

	#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
☐	1	LogID	int(20)			No	None	AUTO_INCREMENT	Change Drop Primary
☐	2	BeaconID	int(10)			No	None		Change Drop Primary
☐	3	UUID	varchar(60)	latin1_swedish_ci		No	None		Change Drop Primary
☐	4	Timestamp	int(10)			No	None		Change Drop Primary
☐	5	Status	int(1)			No	None		Change Drop Primary

11.3 Mobile application

Here I will discuss how I implemented the mobile application. I followed the design section very strictly when implementing the mobile application. This was to ensure all of the system requirements were met. This resulted in the mobile application being similar to the prototypes developed in phase one.

I created the mobile application using Cordova. To create the Cordova project I used the command line interface (Cordova, 2015). See Code listing 7.

Code listing 7: Creating the mobile application project using Cordova

```
v1201-0a984ec9:~ Stu$ cordova create alert com.hoffi.alert Alert
Creating a new cordova project with name "Alert" and id "com.hoffi.alert" at loc
ation "/Users/Stu/alert"
v1201-0a984ec9:~ Stu$
```

I then added the iOS platform using the cordova “platform add ios” command. To add plugins to the Cordova project I used the “cordova plugin add #pluginname#” command. I then ran the Cordova “build” command to generate all the project files. I updated the project to use my own HTML5 documents by navigating to the “alert/platforms/ios/www” directory and inserting them.

I organised the mobile application using a two-tier system inside the www folder so the main files such as index.html, cordova.js and cordova_plugin.js are located in the root directory. Then the audio, CSS, images, JavaScript, PHP and plugin files are located in their own sub folders. This resulted in everything being easy to find and organised.

To navigate to different pages within the mobile application I will wrap HTML page content in page DIV's and give each page a unique ID. The open page will be displayed and the other pages will be hidden. To change pages a JavaScript function will be called that will parse through the next page's ID. All pages will then be hidden using their class "pages" and the next page will be displayed using its unique ID.

11.3.1 Home page

The home page has a button which allows users to open the story page using the changepage function and start the interactive experience.

11.3.2 Story page

Most of the functionality inside the mobile application will take place on the story page.

To get iBeacon details the onBeaconsRanged function mentioned in 6.5.2 section. This gets iBeacon information using the Estimote Cordova API. It then loops over each iBeacon and saves their major, minor, RSSI, distance and colour into an array.

If there is a problem with the BLE on the smartphone the error callback function will be run and an error message will be displayed telling the user the problem in a user-friendly way. This will either be because the smartphone does not support BLE or the Bluetooth on the phone is turned off. One of the non-functional requirements was for the system to be forgiving and this contributes towards that.

These are then looped over using a for loop to append the details to the story page. This for loop is also mentioned in section 6.5.2.

A reverse "for loop" loops over the array of iBeacons to check if user is inside the radius of any iBeacons. It works by starting at the iBeacon located furthest away (end of the array) and finishes at the closest iBeacon (start of the array). A reverse for loop shown in Code listing 8 is used instead of a normal for loop which is described in 6.2.2 so the closest iBeacon content is

displayed to the user. This fixes the error I discovered with the phase one system where if a user was inside two iBeacon radii the closer iBeacon's content would be overwritten by the iBeacon further away.

Code listing 8: Reverse for loop

If there are no iBeacons available and therefore none in the array a user-friendly message will be output telling the user No iBeacons can be found.

```
for( var i=array.length-1; i>=0; i--)
{
```

Inside this reverse for loop the user's UUID is converted to a hash value. To do this I used blueimp's md5 JavaScript library (Blueimp, 2014). See Code listing 9. I used this library, because it has a lot of documentation, it is easy to use and efficient. To install it I included the md5.min.js file in my index file. This ensures complete anonymity and allows each user to still be uniquely identified which was one of the main functional requirements.

Code listing 9: Converting the UUID to a has value

```
//uniquely identify users
var uuid = device.uuid;
uuid = md5(uuid);
```

After the hash value is calculate the distance value to each iBeacon is checked against the different radii for each iBeacon to determine if the user is inside the radii or not. For iBeacons 1, 2 and 3 the code is similar to the 6.3.2 method, but has been enhanced so a pushData method is called to send data to the server when the user enters or exits the 2 metre outer radius. See Code listing 10. For iBeacon 4 the 6.2.2 method is used, similar to above a pushData method is called then the user enters or exits the 1 metre radius. If the user doesn't enter or exit the radius the next iBeacon will be checked in the reverse for loop.

Code listing 10: Calling the push data function

```
pushData(beaconid, uuid, timestamp, status);    //run ajax function to send data to the datalog
```

The beaconid value is then created by concatenating the major and minor values. The pushData function shown in Code listing 11 will parse through the beaconid, hashed uuid, current timestamp and enter or exit status.

Code listing 11: Push data function

```
function pushData(beaconid, uuid, timestamp, status)
{
    if(networkConnection === 1) //network connection available
    {
        var localData;  //variable to store data from the local storage
        networkOnline();    //make sure the error message is hidden
        localData = getAllFromLocalDatabase();  //get local db data
        if(localData.length > 0)    //loop over each set of data in local storag
        {
            for (var i = 0; i<localData.length; i++)    //loop over each record in local storage seperating the value
            {
                var localDataBeaconID = localData[i][0];    //beaconid from local storage
                var localDataUUID = localData[i][1];        //UUID from local storage
                var localDataTimestamp = localData[i][2];   //timestamp from local storage
                var localDataStatus = localData[i][3];      //status from local storage
                //send previous offline or failed data to the server
                pushToDatalog(localDataBeaconID, localDataUUID, localDataTimestamp, localDataStatus);
            }
            removeAllFromLocalDatabase();   //emtoy local database as everything has been sent to the server
        }
        pushToDatalog(beaconid, uuid, timestamp, status);   //push the current data to the server
    }
    else if(networkConnection === 0)    //no network available
    {
        networkOffline();   //make sure error message is displayed
        addToLocalDatabase(beaconid, uuid, timestamp, status);  //add data to local database
    }
}
```

For the data to be pushed to the server a network connection must be available. To check this I added two event listeners to the network connection, one for online, one for offline. These will trigger callback function if the network connection changes which will update the networkConnection status to 1 if available and 0 if unavailable. If there is an error the network error message will be displayed. To access network details the Cordova network information plugin needed to be installed. Like with the BLE error trapping this contributes towards the functional requirement of making the system forgiving.

A check against the network connection variable is made inside the pushData function before attempting to send data to the server.

If no network connection is available the data will be stored in local storage using the addToLocalDatabase function. See Code listing 12. This data will be stored locally until a network connection is next available.

Code listing 12: Add data to local storage function

```
function addToLocalDatabase(beaconid, uuid, timestamp, status){
    var len = window.localStorage.length;   //number of files in local storage
    var key = len;  //position to insert the values
    var obj = JSON.stringify([beaconid, uuid, timestamp, status]);  //convert the values to a JSON string
    window.localStorage.setItem(key, obj);  //insert JSON string into local storage
}
```

If a network connection is available first all the data will be retrieved from local storage using the getAllFromLocalDatabase function. If there is no data in local storage the current beaconid, hashed uuid, current timestamp and enter or exit status will be sent to the server through the pushToDataLog function. See Code listing 13. This data will be looped over and parsed through the pushToDataLog function that is used to send data to the server. Once each set of data in local storage has been looped and set to the server it will be removed using the removeAllFromLocalDatabase function as it is no longer needed. Once the local storage is empty the current beaconid, hashed uuid, current timestamp and enter or exit status will be sent to the server through the pushToDataLog function.

Code listing 13: Mobile application AJAX GET request

```
function pushToDatalog(beaconid, uuid, timestamp, status)
{
    jQuery.ajax({
        type: "GET", // HTTP method POST or GET
        url: "http://redmanexperiences.co.uk/ips/pushdata.php", //Where to make Ajax calls
        data : {beaconid: beaconid, uuid: uuid, timestamp: timestamp, status:status},
        dataType:"jsonp", // Data type, HTML, json etc.
        crossDomain: true,  // Data is being sent to another server
        contentType: 'application/json',     // JSON content
        success:function(response){ //confirmation data has been sent
            console.log(response.state);
        },
        error:function (xhr, ajaxOptions, thrownError){ //if there is an error
            console.log(xhr);
            console.log(ajaxOptions);
            console.log(thrownError);
            addToLocalDatabase(beaconid, uuid, timestamp, status);  //add data to local database
        }
    });
}
```

This data is sent using an AJAX GET request to the pushdata.php file located on the redman server. At first I struggled to get this AJAX request working due to the cross-domain issue.

Code listing 14: Mobile application config.xml access origin

```
<access origin="http://redmanexperiences.co.uk" />
```

Finally after lots of research I realised that I hadn't given access to the server domain redmanexperiences.co.uk in the mobile application config.xml file. See Code listing 14. This is essential so external servers are not blocked when communicating with the mobile application.

To establish a connection and insert the beaconid, uuid, timestamp and status into the database a PHP file would be used. This is required as this cannot be done securely through

JavaScript from a mobile application using Cordova. To interact with the PHP file I set the URL from the AJAX GET request to the pushdata.php file located on the redman server. I then formatted the data so it was a suitable JSONP datatype.

In the pushdata.php file located on the server I first stored the beaconid, uuid, timestamp and status values as variables using the $_REQUEST method. I then defined the database server details including the IP address, username, password and database name. To check data was correctly sent to the server I verified none of the variables were empty.

If a database connection failed an error message containing the problem would be sent back as a response to the AJAX request. If the database connection was successful the prepared statement was then created. See Code listing 15. I used a prepared SQL statement as it reduces parsing time as the query preparation is only done once and only parameters are sent to the database each time which saves bandwidth. Prepared statements are also effective against SQL injections. The prepared statement inserts the beaconid, uuid, timestamp and status data into the datalog table.

Code listing 15: MySQL prepared statement to insert data into the database

```
//prepared sql statement to insert the data into the datalog table
$stmt = $mysqli->prepare("INSERT INTO datalog(BeaconID,UUID,Timestamp,Status) VALUES (?, ?, ?, ?)");
```

I then bind the beaconid, uuid, timestamp and status values to the prepared statement. See Code listing 16.

Code listing 16: Bind values to the prepared statement

```
$bind = mysqli_stmt_bind_param($stmt, "isii", $beaconid, $uuid, $timestamp, $status);
```

Code listing 17: Execute the prepared statement

```
$exec = mysqli_stmt_execute($stmt);
```

Before executing the statement using the execute command. See Code listing 17.

I have also included error trapping in case the statement, binding, or execution fails. This will return an error message to the AJAX request.

If everything is successful and data is inserted into the database a successful response will be returned to the AJAX request. See Code listing 18.

Code listing 18: AJAX callback value

```
echo $_GET['callback'].'('.$arr.')';    //something needs to be returned to the AJAX call as confirmation
```

After the callback the prepared statement and connection are closed.
The user is not notified the data has been sent to the server as this would result in unnecessary notifications every few seconds.

Once this process is completed the reverse for loop continues onto the next iBeacon and repeats the same process until all iBeacons have been checked. Once all the iBeacons have been checked the mobile application listens out for iBeacons again and when one of the iBeacons broadcast information the onBeaconRanged function will run triggering this process all over again.

Overall the implementation ran smoothly with the only big issue encountered being the cross-domain issue with mobile applications mentioned above. The data flow and sequence diagrams were particularly useful to refer back to so I could make sure all parts of the system had been implemented.

11.4 System management interface

Here I will discuss how I implemented the system management interface. As with the mobile application implementation I followed the design section when implementing the system management interface to ensure all of the system requirements were met.

Similarly to the mobile application I organised the system management interface using a two-tier system so the main index.html file is located in the root directory. Then the audio, CSS, images, JavaScript, PHP and video files are located in their own sub folders. This resulted in everything being easy to find and organised.

To navigate to different pages I used the same navigation changepage function used in the mobile application.

11.4.1 Home page

The purpose of the home page is to navigate between the four different pages in the system management interface. To do this I created four different buttons leading to the view all data, behaviour data, monitor data and horror frame pages that trigger the changepage function on click.

11.4.2 View all data page

The view all data page works by querying all the data from the database. To do this I run an AJAX GET request similarly to the mobile application, but the purpose of this request is to retrieve data rather than send data. See Code listing 19. I used a JavaScript and AJAX solution rather than PHP to enable portability with the system management interface, as it will only be run on local machines. I already had the majority of the code in place from the mobile application so this would save me time. The AJAX GET request runs when the view all data page loads or when the refresh button is pressed.

Code listing 19: System management interface AJAX GET request

```
type: "GET", // HTTP method POST or GET
url: "http://redmanexperiences.co.uk/ips/getalldata.php", //Where to make Ajax calls
dataType:"jsonp", // Data type, HTML, json etc.
crossDomain: true,
contentType: 'application/json',
```

The AJAX GET request doesn't send any data, but still uses the JSONP data type so it works cross-domain. The AJAX call is sent to the getalldata.php file that is located on the redman server.

In the getalldata.php file like with the pushdata.php file in the mobile application I defined the database server details. I then attempted to connect to the database, if there was an error it would be sent back to the system management interface via the AJAX callback method. If connecting to the database was successful a prepared select SQL statement will be created to select everything from the database. See Code listing 20.

Code listing 20: Get all data select query

```
if($stmt = $mysqli->prepare("SELECT LogID, BeaconID, UUID, Timestamp, Status FROM datalog ORDER BY Timestamp DESC"))
{
    $stmt->execute();
    $stmt->bind_result($logid, $beaconid, $uuid, $timestamp, $status);
    $data = array();
    $i = 0;
    while ($stmt->fetch())
    {
        $data[$i][0] = $logid;
        $data[$i][1] = $beaconid;
        $data[$i][2] = $uuid;
        $data[$i][3] = $timestamp;
        $data[$i][4] = $status;
        $stmt->store_result();
        if($stmt2 = $mysqli->prepare("SELECT RoomID FROM beacons WHERE BeaconID = ?"))
        {
            $stmt2->bind_param("i", $beaconid);
            $stmt2->execute();
            $stmt2->bind_result($roomid);
            while ($stmt2->fetch())
            {
                $data[$i][5] = $roomid;
                $stmt2->store_result();
                if($stmt3 = $mysqli->prepare("SELECT Name, LocationID FROM rooms WHERE RoomID = ?"))
                {
                    $stmt3->bind_param("i", $roomid);
                    $stmt3->execute();
                    $stmt3->bind_result($roomname, $locationid);
                    while ($stmt3->fetch())
                    {
                        $data[$i][6] = $roomname;
                        $data[$i][7] = $locationid;
                        $stmt3->store_result();
                        if($stmt4 = $mysqli->prepare("SELECT Name, Longitude, Latitude FROM locations WHERE LocationID = ?"))
                        {
                            $stmt4->bind_param("i", $locationid);
                            $stmt4->execute();
                            $stmt4->bind_result($locationname, $longitude, $latitude);
                            while ($stmt4->fetch())
                            {
                                $data[$i][8] = $locationname;
                                $data[$i][9] = $longitude;
                                $data[$i][10] = $latitude;
                                $stmt4->store_result();
```

The select statement first selects LogID, BeaconID, UUID, Timestamp and Status from the datalog table. For each record returned it uses the BeaconID to find the relevant RoomID in the beacons table. It then uses the RoomID to retrieve the correct room Name from the rooms table. Then using the RoomID it finds the matching LocationID from the rooms table. This locationID is then used in the locations table to find the location Name, Latitude and Longitude. Each of the values are inserted into the data array. By the end of the query the LogID, BeaconID, UUID, Timestamp, Status, RoomID, room Name, LocationID, location Name, Latitude and Longitude will be stored in an array for each record. This array is then converted to JSON and returned using the AJAX callback method. See Code listing 21

Code listing 21: Get all data AJAX callback

```
$json = json_encode($data);
echo $_GET['callback'].'('.$json.')';
```

If there is a problem with any of the prepared statements an error message will be returned via

the AJAX callback.

After the callback the prepared statement and connection to the database are closed.

The system management interface AJAX success or function will trigger depending if communicating with the getalldata.php was successful or not. If error function triggers the error message will be output.

If the success function is called first the length of the response will be checked to determine if the database was empty or not. If the database was empty a message will be output notifying the user.

If there is at least one piece of data in the response the data will be looped over. Most of the data will already be in a suitable format to display to the user, but the timestamp and status will need to be converted to a user-friendly form. To convert the timestamp I used the Date object. See Code listing 22.

Code listing 22: Converting a timestamp to a UK date and time

```
var timestamp = response[i][3];
var datetime = new Date(timestamp*1000);
var months = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'];
var year = datetime.getFullYear();
var month = months[datetime.getMonth()];
var date = datetime.getDate();
var hour = datetime.getHours();
var min = datetime.getMinutes() < 10 ? '0' + datetime.getMinutes() : datetime.getMinutes();
var sec = datetime.getSeconds() < 10 ? '0' + datetime.getSeconds() : datetime.getSeconds();
date = date + ' ' + month + ' ' + year;
time = hour + ':' + min + ':' + sec;
```

To change the status from 1 or 0 to enter or exit I used an if statement.

To display the data to the user I used a table so the different records and columns could be clearly highlighted. For each record I appended the data to the display-all-data div.

11.4.3 Monitor page

The monitor page is similar to the view all data page, but queries the logdata table every second. To make it query every second I set a variable to true if they are on the monitor page and false if they aren't. This variable is checked every second using a JavaScript setInterval function, if it is true the query will be made to the database using an AJAX GET like the view all data method.

The AJAX GET method is the same, but calls the continuouslyquerydata.php file instead.

The continuouslyquerydata.php file connects to the database like the getalldata.php file, but runs a different query as I only need data from the most recent record for each user from the datalog table. To do this I used an inner join. See Code listing 23.

Code listing 23: Select query using an inner join

```
if($stmt = $mysqli->prepare("SELECT t1.* FROM datalog AS t1 WHERE t1.LogID =
    (SELECT t2.LogID FROM datalog AS t2
        WHERE t2.UUID = t1.UUID
        ORDER BY t2.Timestamp DESC
        Limit 1)"))
{
```

The data selected includes LogID, BeaconID, UUID, Timestamp and Status which are all from the datalog table. As with the getalldata.php file the data is sent back through the AJAX callback function.

Once the data is returned the same check as the view all data is made to check if data is returned. If data is returned it will loop over each record checking to see if the user has entered the radius of any iBeacon in the last 5 minutes. Only data inside 5 minutes is used as the longest audios and video files used in the system are just under 5 minutes. So to keep the system querying consistently the last minutes of data are used throughout the system management process when queries are repeated every second.

The check to determine whether a user is inside or outside the iBeacon radius is similar to the mobile application iBeacon radius check, but checks against the status instead as shown in Code listing 24 below.

Code listing 24: Monitor page content delivery check

```
if(timestamp > timeDifference)  //only use data from the last 'timedifference' minutes
{
    if(beaconid === 443331)
    {
        if(status === "Enter" && playContent1 === 0)
        {
            playContent1 = 1;
            console.log("user entered - play content1 - baby video");
            $('#monitor-video1').attr('src', "https://www.youtube.com/embed/8t8jo4cMGIU?autoplay=1");
            $('#monitor-video1').show();
            $('#monitor-video1').removeClass(".hide");
            $('#monitor-description').text("Enjoy the beacon1 baby video :)");
        }
        else if(status === "Exit" && playContent1 === 1)
        {
            playContent1 = 0;
            $('#monitor-description').text("No user is within 2 metres of the beacons");
        }
    }
```

If the user is inside the radius and the content isn't previously playing the relevant content video will be displayed and played to the user through the system management interface monitor page. If the user has just exited the radius and the playContent1 variable will be set to 1 (false).

After each record returned through the AJAX callback has been looped and checked each playContent variable set to false will be hidden and paused.

11.4.4 Behaviour page

The behaviour page uses nearly the same method as the monitor page to query the database, but has a slightly different query. It selects the LogID, BeaconID, UUID, Timestamp and Status for each record in the datalog table in descending order. This query is also run every second if the user is on the behaviour page.

The content delivery method with the behaviour is quite similar to the monitor pages, but slightly more complex. Beacon3 content is only delivered if the user has previously viewed Beacon 1 and 2 content. Beacon2 content is only delivered if the user has viewed Beacon1 content.

To make these checks I use JQuery cookies. I decided to use carhartl's jquery-cookie library (Carhartl, 2011), a simple and lightweight system for reading, writing and deleting of cookies. Each cookie ID will be made up of the users UUID and the BeaconID. So each interaction between a user and iBeacon will be unique.

Code listing 25: Initialise cookie

```
var name = uuid.concat(beaconid);
if($.cookie(name) === undefined)
{
    $.cookie(name, 0);
}
```

A cookie will be made at the start of each loop for the current user UUID and iBeacon and set to 0 (false) if it hasn't been initialised. See Code listing 25.

Code listing 26: Update cookie value

```
$.cookie(checkBeacon1Visited, 1);
```

If a user is inside the iBeacon1 radius the iBeacon1 video will play and the relevant cookie value will be updated to 1. See Code listing 26.

Code listing 27: Check cookie values

```
var beacon1Status = $.cookie(checkBeacon1Visited);
var beacon2Status = $.cookie(checkBeacon2Visited);
if(beacon1Status == 1 && beacon2Status == 1)
{
```

When entering iBeacons 2 and 3 a check against specific cookies will be made to see if the user has previously viewed the content for the relevant iBeacons. See Code listing 27.

If this is true the relevant video is displayed to the user on the behaviour page, but if it is false the user will be notified they haven't visited the other iBeacons.

11.4.5 Horror frame page

The horror frame page content uses the same AJAX GET method to query the database as the monitor page, but uses a much more complex query. The query uses a left outer join on the UUID so only the most recent record for each user inside the last 60 seconds is returned. See Code listing 28. To get the last 60 seconds I get the current server timestamp and then subtract 1 minute. In the query the timestamp value must be greater than this value. This query is also run every second if the user is on the behaviour page.

Code listing 28: Left outer join select query

```
$stmt = $mysqli->prepare("SELECT t1.LogID, t1.BeaconID, t1.UUID, t1.Timestamp, t1.Status
    FROM datalog AS t1
    LEFT OUTER JOIN datalog AS t2
        ON t1.UUID = t2.UUID
            AND (t1.Timestamp < t2.Timestamp
                OR (t1.Timestamp = t2.Timestamp AND t1.LogID < t2.LogID))
    WHERE t2.UUID IS NULL AND t1.Timestamp >= ?");
$stmt->bind_param('i', $timerange);
```

The selected LogID, BeaconID, UUID, Timestamp and Status is then returned to the system interface through the AJAX callback like the monitor page method.

The content delivery method is simpler than the monitor and behaviour systems. Each piece of data retrieved using the query will be looped over and checked to see if they have entered iBeacon4 in the last 60 seconds. If the user has entered iBeacon4 in the last 60 seconds the count will be incremented.

Once each piece of data has been looped over the count will be checked. If the count is 1 the video will play on a loop. If the count it 0 or more than one the video will stop and return to the start.

11.5 Environment

I used the environment design as a template when positioning the different parts of the system, but due to the room size and lack of computers there were a few changes to the environment.

iBeacons 1,2 and 3 were placed around 5-6 metres apart at chest height (approx 1.5 metres). The horror story iBeacon was placed quite low down at around 50cm as the table was low and to place it any higher would have interfered with the screen projection.

To display the system management interface content I had to cycle between the different monitor and behaviour data pages due to only having access to one monitor.

The view all data page was displayed on my laptop so the Yellobrick employee could view all the user behaviour data.

Figure 47: Horror frame environment picture

The horror frame screen was projected from the laptop onto a white picture frame. See Figure 47.

12 Description of Final System Phase Two: Yellobrick Prototype

Here I will talk through each part of the system explaining how it works and how it contributes to the goals of the system.

12.1 Mobile application

The purpose of the mobile application was to interact with the iBeacons so the user's position could be calculated and send data to the server when specific location conditions are met. The application also needed to tell the user how far away from each important location they are and notify them if they entered or exited a specific area.

12.1.1 Home page

Figure 48: Mobile application home page

The mobile application home page needed to provide navigation to the story page so the user could choose when they wanted to start the interactive experience. This was achieved through the light blue story button. The user can return to the home page by pressing "home" on the story page. See Figure 48.

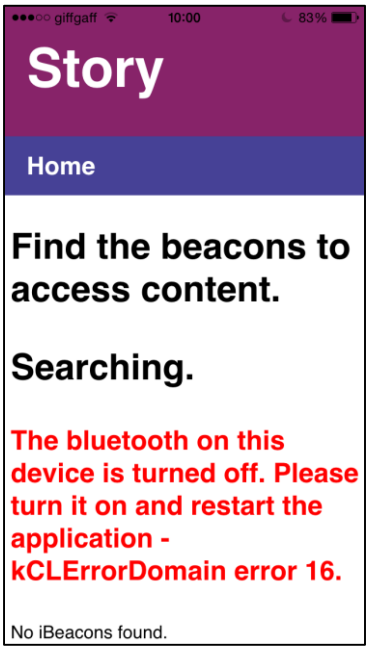
12.1.2 Story page

The story page starts the interactive experience so the user could communicate with the iBeacons. This would play a vital part in delivering a working system as the back-end needed to determine the position of the user and send data to the server when the user entered or exited a specific area so the system management interface could view and use the user's behaviour. One of the main requirements was for the user to be uniquely identified anonymously. I ensured this by changing the device UUID to a hash value before sending the data to the server. It also needed to deliver iBeacon information and content to the user through the mobile application interface front-end.

Once the page had loaded it starts searching for iBeacons.

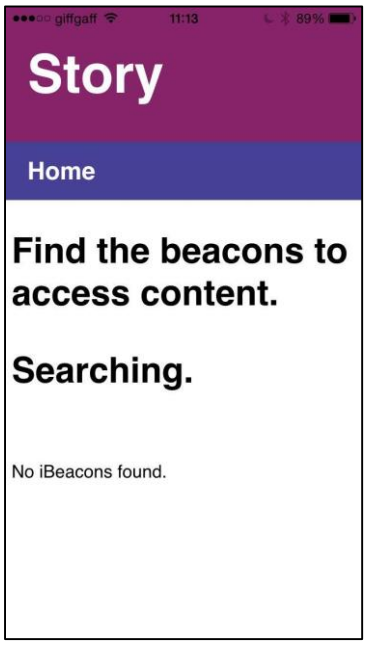
One of the non-functional requirements was for the system to be user-friendly. To contribute towards a user-friendly system it would need to be forgiving and intuitive so the user's know when something is wrong. To ensure this I made sure users are notified when there is a problem with the system.

Figure 49: Story page when Bluetooth is turned off



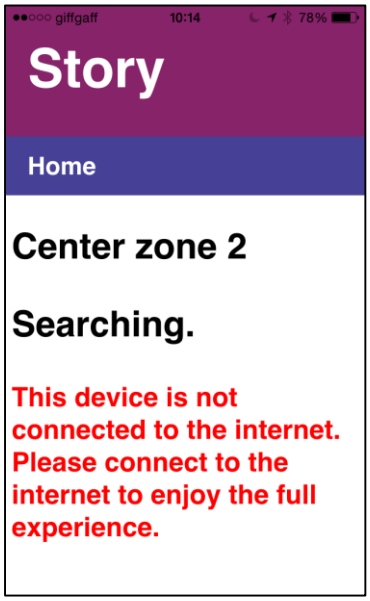
If the phone doesn't support Bluetooth LE or Bluetooth is turned off the user is notified with a message saying "This device does not support Bluetooth Low Energy (BLE) - kCLErrorDomain error 17." or "The bluetooth on this device is turned off. Please turn it on and restart the application - kCLErrorDomain error 16." See Figure 49.

Figure 50: Story page no iBeacon available or within range



If no iBeacons are found the user is notified with a message telling them “No iBeacons found” See Figure 50.

Figure 51: Story page no network connection available



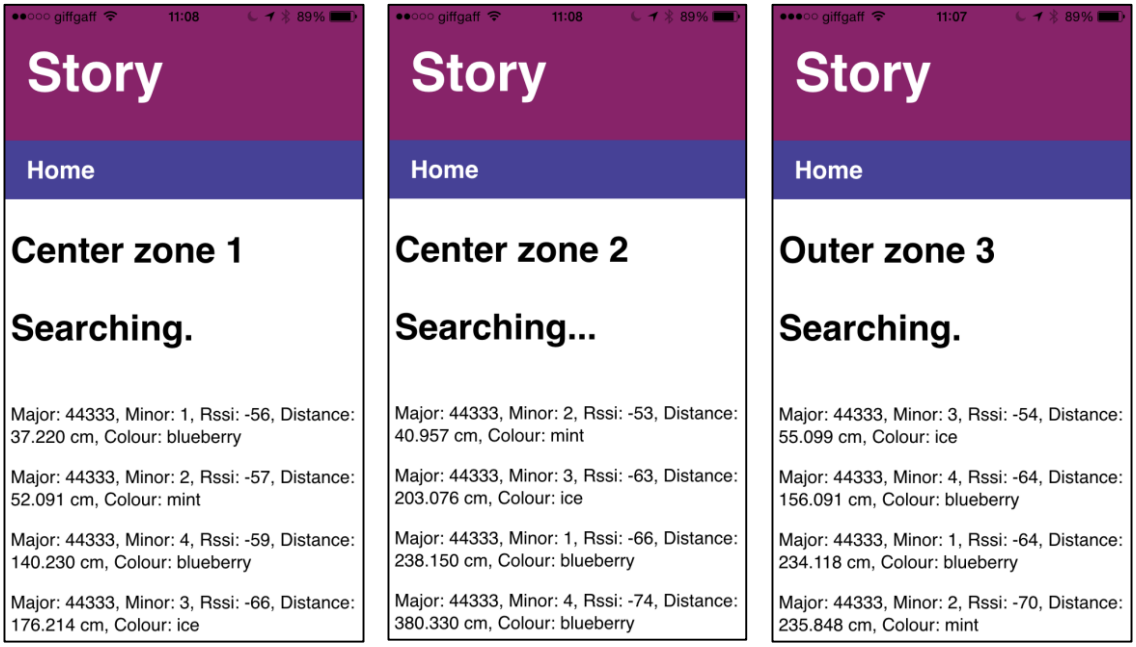
A network connection is required when sending data to the database. To ensure the system doesn’t fall over if no network connection is available I check the network connection. If there is a problem with the network connection the data is stored in local storage and a message is output to the user saying “This device is not connected to the internet please connect to the internet to enjoy the full experience”. See Figure 51.

If there are no problems with the system and iBeacons are detected their major, minor, RSSI, distance and colour details are output on the screen.

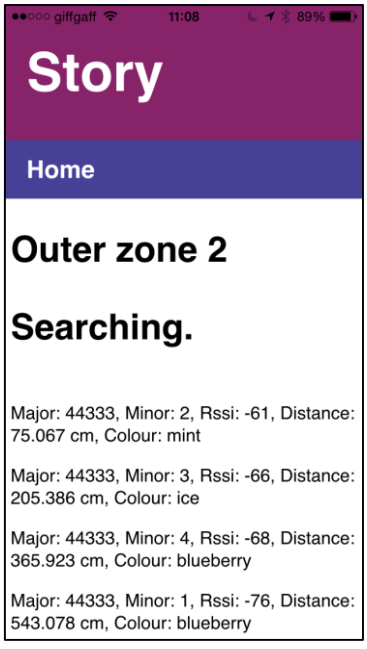
One of the main requirements was to send user data to a remote server when a network connection is available and the user enters or exits the radius. To do this a check is made for each iBeacon to see if the user is inside the radius. If the user enters or exits the radius the beaconid, hashed uuid, current timestamp data is sent to the server. For iBeacons 1,2 and 3

zones are used, but the data is only sent if the user enters and exits the outer zone of 2 metres. iBeacon4 uses a radius of 100cm.

Figure 52: Screenshots of the story page detecting multiple iBeacons



Content is delivered to the user via text and audio. The user is notified through text when they are inside the inner zone, outer zone or exit the outer zone. See Figure 52. Audio content is played quietly through the mobile application when they enter the outer radius of iBeacons 1,2 and 3 and gradually gets louder the closer the user gets to the iBeacon. Then when they enter the inner radius the audio volume is turned up to full. For iBeacon4 it is when they enter or exit the 100cm radius.

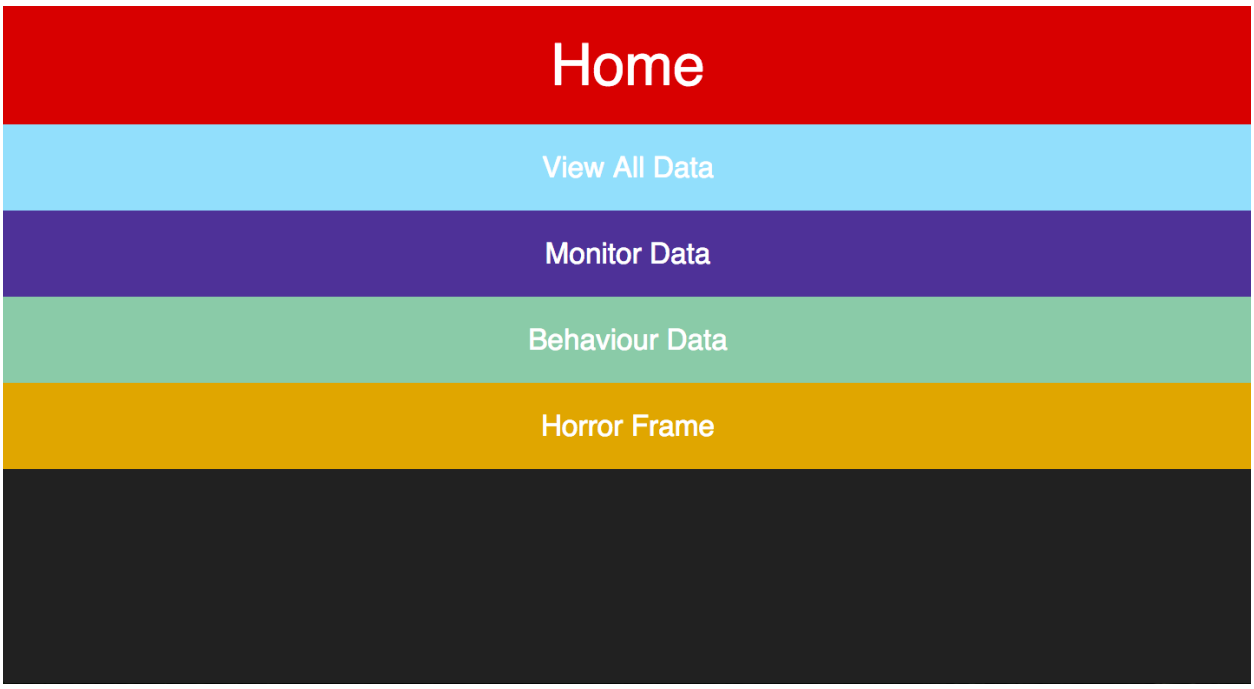


12.2 System management interface

The system management interface must allows Yellobrick employees to track user behaviour and set up content on monitors. The content needs to vary depending on the mobile application user's physical location. Some content needs to trigger when users entered specific locations and other content would only trigger when users visited the locations in a specific order. Another part of the interface would involve a user walking close to a painting triggering a video so the painting starts moving and then if another user gets close to the painting it will stop moving.

12.2.1 Home page

Figure 53: System management interface home page

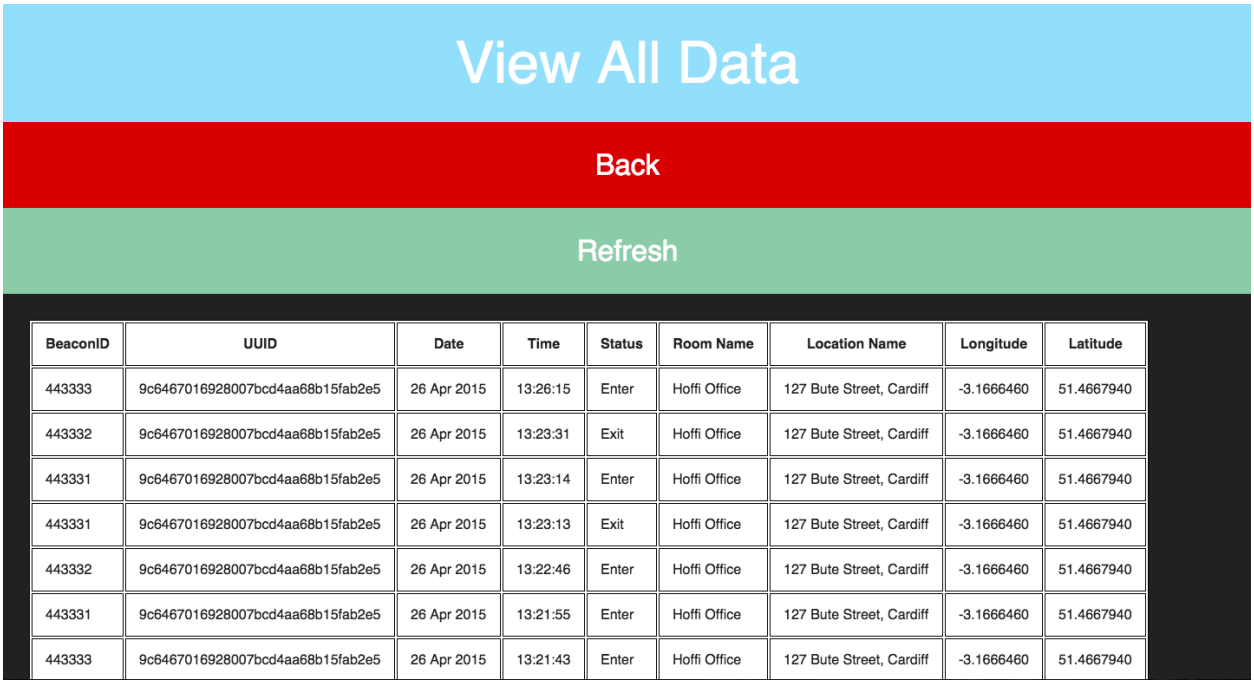


The system management interface home page needs to provide navigation the different pages available in the system management interface process. This was achieved by creating buttons that navigated to each of the pages in the system and then on the landing pages having a back button. See Figure 53.

12.2.2 View all data page

The view all data page must allow Yellobrick employees to view the entire system history and monitor user "dwell times". This was one of the main functional requirements the system had to do. I achieved this by selecting all of the data from the database and displaying it in a table. See Figure 54. From this table the employee can see which iBeacon the user is interacting with, uniquely identify the user using the UUID, view the date and time of the interaction. They can also see if the user entered or exited the specific iBeacon and the location details of the experience.

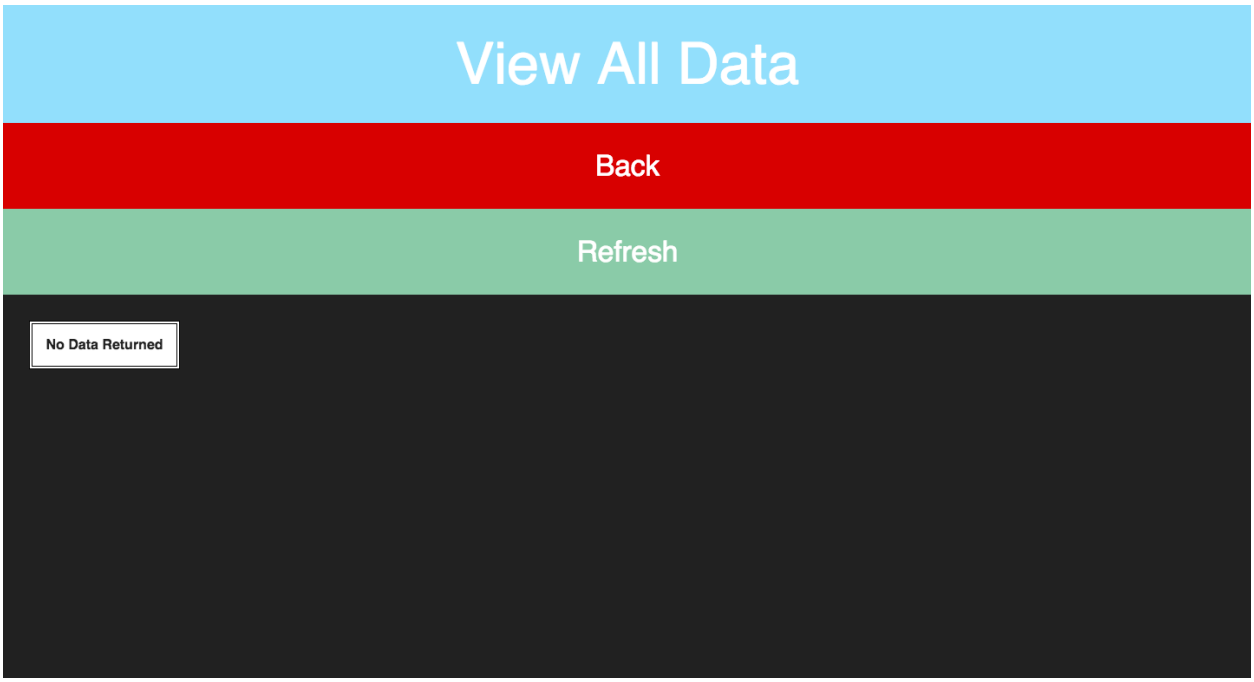
Figure 54: View all data page displaying all user data from database



BeaconID	UUID	Date	Time	Status	Room Name	Location Name	Longitude	Latitude
443333	9c6467016928007bcd4aa68b15fab2e5	26 Apr 2015	13:26:15	Enter	Hoffi Office	127 Bute Street, Cardiff	-3.1666460	51.4667940
443332	9c6467016928007bcd4aa68b15fab2e5	26 Apr 2015	13:23:31	Exit	Hoffi Office	127 Bute Street, Cardiff	-3.1666460	51.4667940
443331	9c6467016928007bcd4aa68b15fab2e5	26 Apr 2015	13:23:14	Enter	Hoffi Office	127 Bute Street, Cardiff	-3.1666460	51.4667940
443331	9c6467016928007bcd4aa68b15fab2e5	26 Apr 2015	13:23:13	Exit	Hoffi Office	127 Bute Street, Cardiff	-3.1666460	51.4667940
443332	9c6467016928007bcd4aa68b15fab2e5	26 Apr 2015	13:22:46	Enter	Hoffi Office	127 Bute Street, Cardiff	-3.1666460	51.4667940
443331	9c6467016928007bcd4aa68b15fab2e5	26 Apr 2015	13:21:55	Enter	Hoffi Office	127 Bute Street, Cardiff	-3.1666460	51.4667940
443333	9c6467016928007bcd4aa68b15fab2e5	26 Apr 2015	13:21:43	Enter	Hoffi Office	127 Bute Street, Cardiff	-3.1666460	51.4667940

To ensure the system was user-friendly in the event no data is in the database or if there is a problem with the database a message will be output to notify the user. If there is no data in the database a "No Data Returned" message is output. See Figure 55. If there is a problem connecting to the database the message "Error connecting to the database. Check your connection" is output.

Figure 55: View all data page returning no data from the database

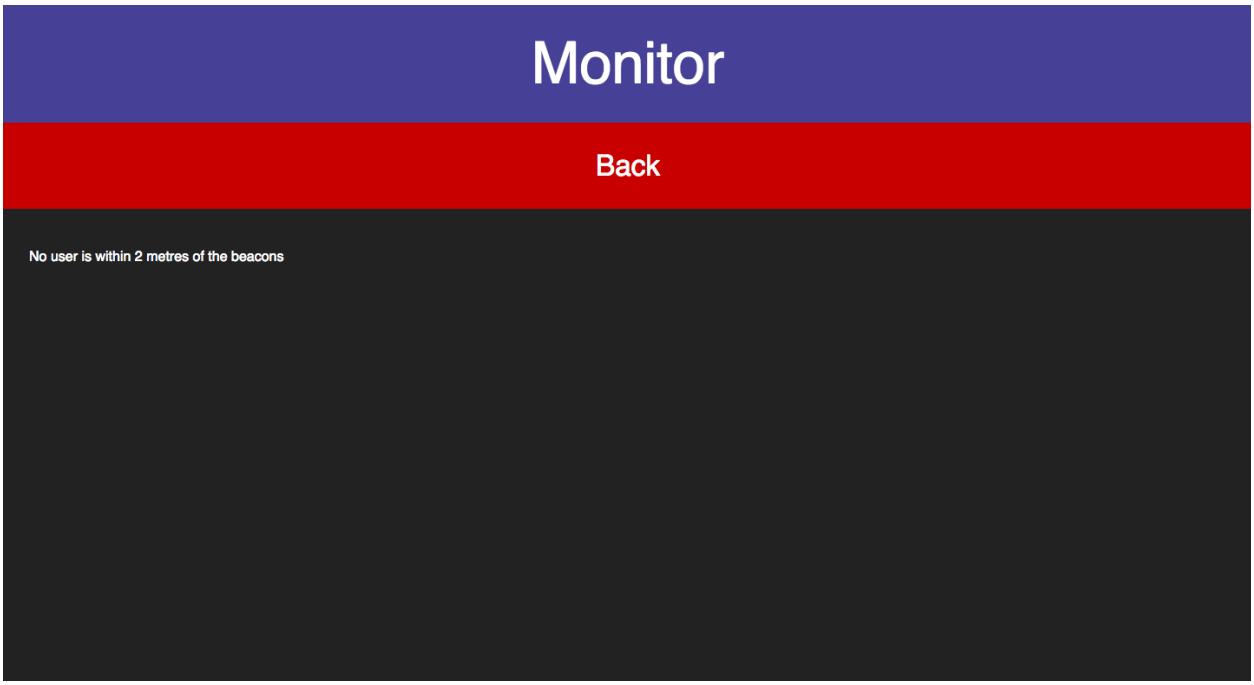


12.2.3 Monitor page:

This page must deliver content to users when they enter iBeacon 1, 2 or 3's radii using live data from the database. This achieves one of the main functional requirements to deliver content to users through the system management interface when users' physical locations are within a certain distance of the iBeacon. The page is loaded onto a monitor by the Yellobrick employee, but viewed by mobile application users.

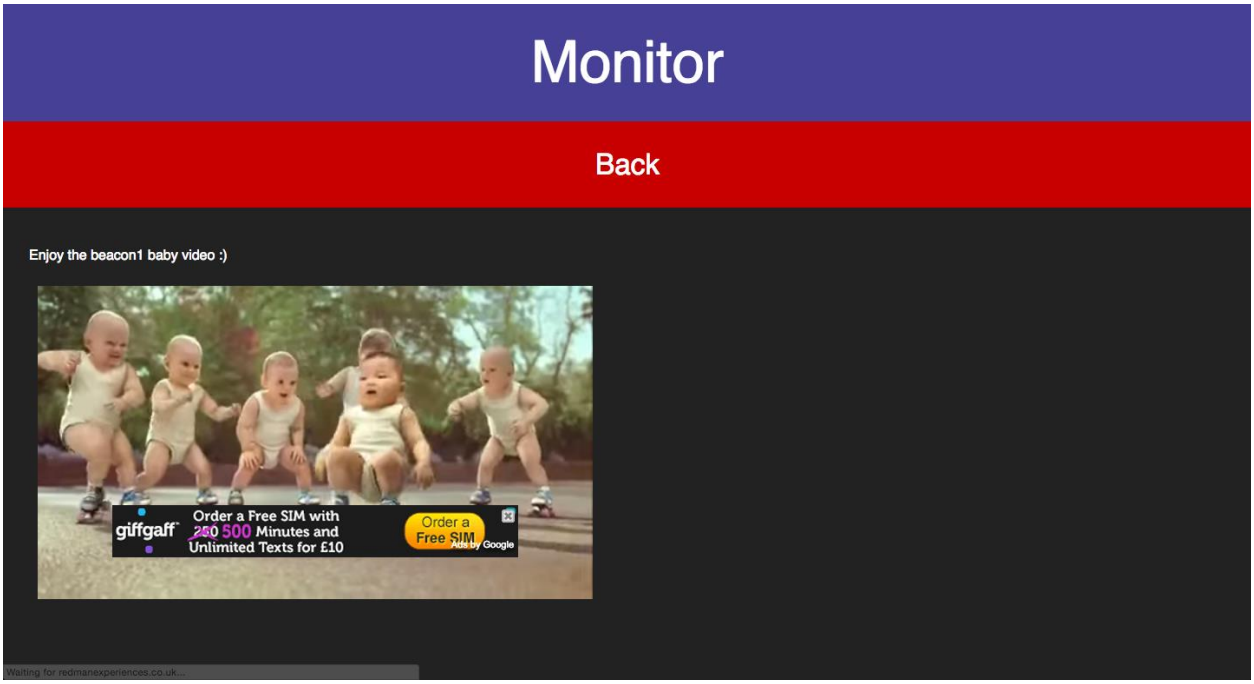
The most recent user activity data inside the last 5 minutes is checked every second to determine which content should be played. If none of the user's last actions were entering an iBeacon radius no videos will be displayed and a message will be output saying, "No user is within 2 metres of the beacons". See Figure 56.

Figure 56: Monitor page when no users have entered any iBeacon radii



If a user's last action was to enter the radius of the iBeacon1 radius a baby video will be displayed and start auto playing. See Figure 57.

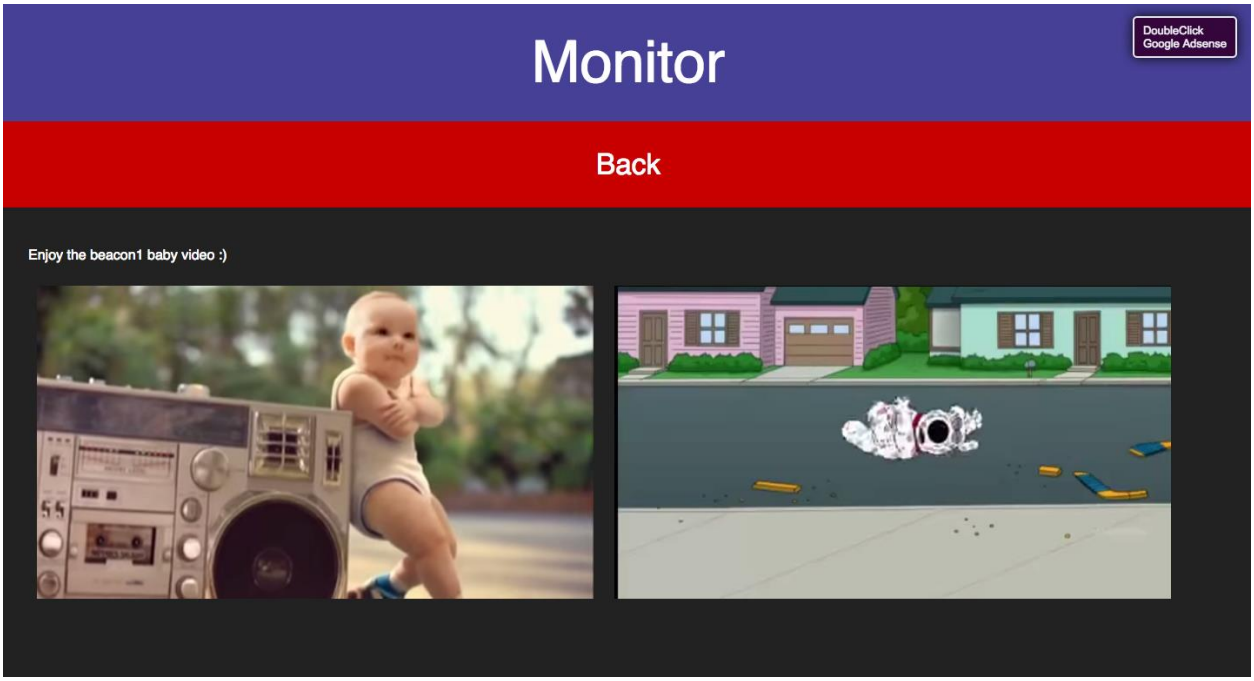
Figure 57: Monitor page when a user is inside the iBeacon1 radius



As with the iBeacon1 check, the same applies to iBeacon 2 and 3. If a user's last action was to enter the radius of iBeacon 2 a surf video will be displayed and start auto playing. If a user's last action was to enter the radius of iBeacon 3, the family guy video will be displayed and start playing.

If there are multiple users who each enter a radius, but for different iBeacons, each of these videos will be displayed and auto played. See Figure 58.

Figure 58: Monitor page when users are inside the iBeacon1 and iBeacon2 radii



12.2.4 Behaviour page:

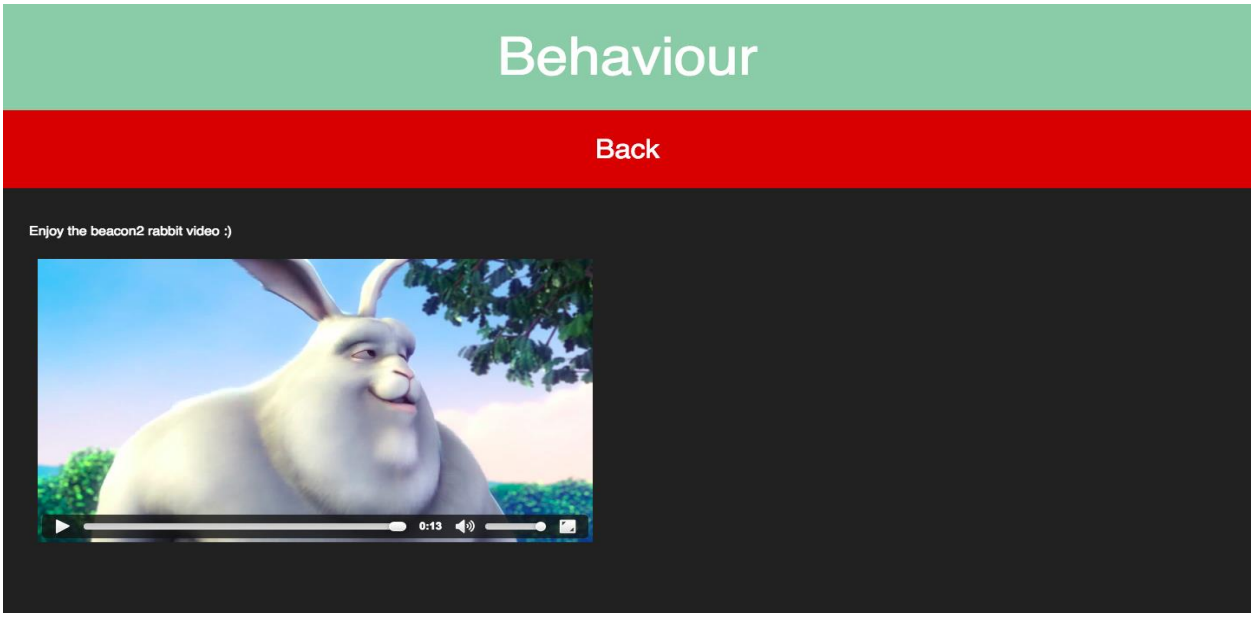
The behaviour page must only display content when a user visits the iBeacons in specific order. Like the monitor page this page is loaded onto a screen by a Yellobrick employee, but viewed by mobile application users. All user behaviour data is retrieved and checked every second to determine which content should be played.

When the user is inside an iBeacon a check is made to ensure the relevant iBeacons have been visited. If the user started the experience and went straight to iBeacon3 without previously visiting iBeacons 1 and 2 the video content would not display and the user would be told via a message to "Visit beacon 1 and 2 first". See Figure 59.

Figure 59: Behaviour page when a user is inside iBeacon3 radius, but has not previously visited iBeacons 1 and 2

For the iBeacon2 video content to be displayed the user needs to have previously visited iBeacon1. IBeacon1 video content activates when a user is inside the radius, as previous iBeacons need to be visited for the content to play. If the preconditions have been met the relevant video content will be displayed and auto played. See Figure 60.

Figure 60: Behaviour page when a user enters iBeacon2 after visiting iBeacon1



12.2.5 Horror frame page:

The horror page must confuse the user in a dark horror scene type setting. Like the monitor page this page is loaded onto a screen by a Yellobrick employee, but viewed by mobile application users. All user behaviour data is retrieved and checked every second to determine whether the horror video should be played or not. The horror video is set to full screen permanently so it looks like a painting.

Figure 61: Horror page video set to full screen and zero or more than one users are inside the iBeacon4 radius

I achieved this by making a check to see how many users were inside the iBeacon. When there are zero or more than one user inside the radius of iBeacon4 the painting is paused at the start. See Figure 61.

Figure 62: Horror page video set to full screen and one user is inside the iBeacon4 radius

If there is one user inside the radius the video will play the video to give a flashing spooky effect confusing the user. See Figure 62.

12.3 Database system

The database must store data remotely so the system management interface and mobile application could interact with the same data. This was one of the main requirements as without this the system management would not be able to access the user behaviour data from

mobile applications. This was achieved as the BeaconID, hashed UUID, current Timestamp and enter or exit Status data from the mobile application is successfully stored in a remote database.

12.4 Environment

The environment worked quite well as iBeacons were placed 5-6 metres apart meaning none of the delivery content overlapped.

Placing the iBeacons at approximately 1.5 metres worked well when users were holding their smartphones towards the iBeacon's, but as expected from my discovery in 7.5 when they faced the other direction the human body caused interference so the distance value fluctuated reducing accuracy.

The limitation of having just one monitor to display the monitor and behaviour sections was frustrating, as the Yellobrick employee has to flick back and for between the two so the user could see the content update on each screen.

The view all data page setup on the laptop worked well as the Yellobrick employee could always access it and refresh to get the latest user behaviour.

The horror frame projected onto a white picture frame and worked really well, but the small picture frame meant users had to get close (within 50 cm) to read the writing displayed in the video.

Overall the environment worked very well, but for future environments separate screens for the monitor and behaviour pages will be needed as mentioned in the recommended environment in 10.4. Also for future environments I would try and project the horror frame video onto a bigger screen.

13 Testing and Evaluation Phase Two: Yellobrick Prototype

Here I will explain the testing strategy I used to evaluate my system. I will then discuss the results of the tests before suggesting other tests that could be carried out to increase confidence in the system. I received feedback from Alison John on behalf of the client Yellobrick upon completion which will also be discussed.

13.1 Testing strategy

This explains how I will test my system.

Testing will be split into three different sections. To test the system functionality I will create test cases to test the system against the functional requirements acceptance criteria. To test the non-functional requirements of the system I will create test procedures. To test the usability of the system I will use the System Usability Scale (SUS) created by John Brooke in 1986 (Brooke, 1986).

13.1.1 System functionality testing

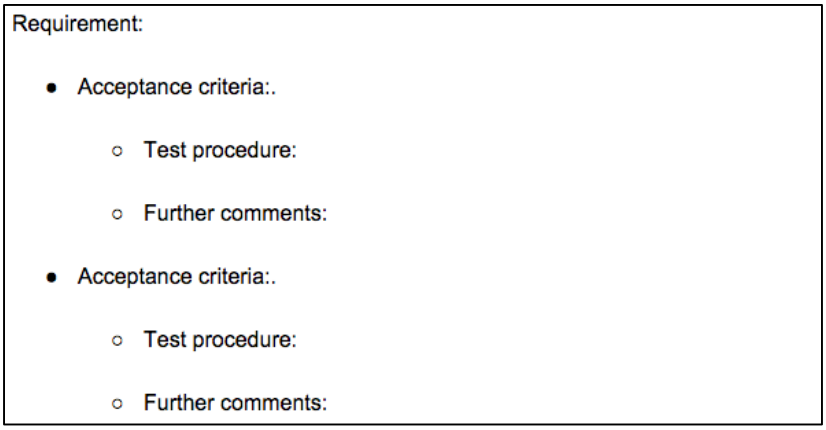
This explains how I will test the system functionality. This will involve writing test cases using the test case template shown in Figure 63 for each acceptance criteria described in the functional requirement section. These test cases will then be completed at a later date discussed in the evaluation. The created test cases and their results are given in Appendix 1.

Figure 63: Test case template

Test Case ID:	**Test Purpose:**		
Environment:			
Preconditions:			
Test Case Steps:			
Step Number:	Procedure:	Expected Response:	Pass/Fail:
Comments:			
Checker:			

13.1.2 Non-functional requirement testing

Figure 64: Non-functional requirement test procedure template



This explains how I will test all non-functional requirements other than usability. To test each requirement I will write a test procedure using the template in Figure 64 to make sure it complies with the acceptance criteria of each requirement. Each of these test procedures and their results are described in Appendix 2.

13.1.3 Usability testing

This tests how user-friendly the system is. In order to evaluate the usability of the system I would need to measure the user satisfaction of using it. For this purpose the system needs to be tested by a sample of users. A study conducted by Jacob Nielson proved the best results come from testing no more than five users (Nielsen, 2000).

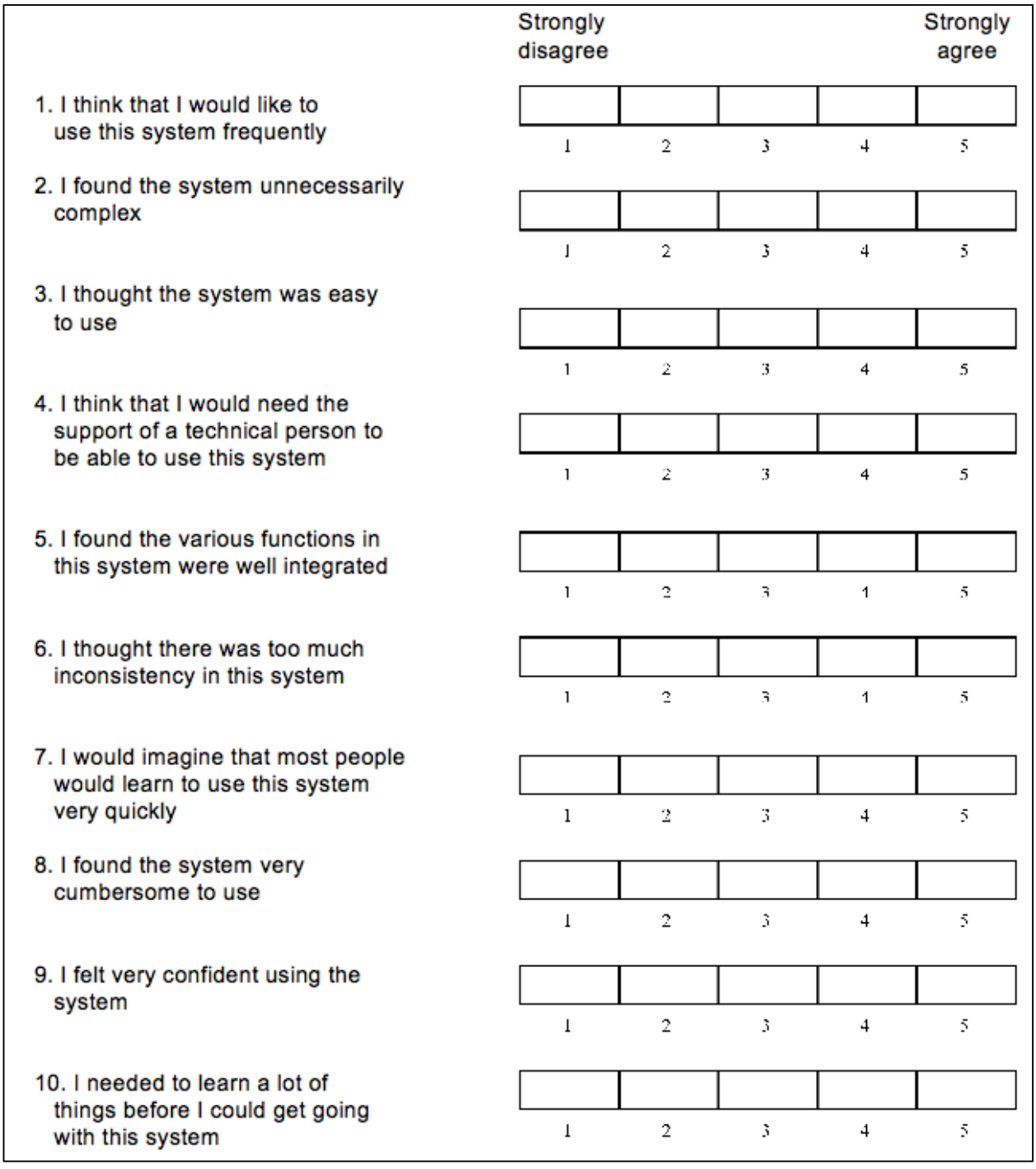
I will use the System Usability Scale (SUS) created by John Brooke in 1986 (Brooke, 1986) to measure the usability of the system. The SU scale consists of 10 questions, answered from strongly agree to strongly disagree. The users are asked to record their response to each item immediately rather than thinking about items for a long time. All items must be checked, if the respondent feels they cannot respond to a particular item they should mark in the centre point of the scale. The SU scale will be handed out to each test user who will be asked to complete the scale for both the mobile application and system management interface.

After five SU scales have been filled in for both the mobile application and system management interface an SUS score will be calculated. To calculate the SUS score, first the score contributions from each item with be added up. Each item's score contribution will range from 0 (strongly disagree) to 4 (strongly agree). For items 1,3,5,7,and 9 the score contribution is the scale position minus 1. For items 2,4,6,8 and 10, the contribution is 5 minus the scale position. After the sum of these scores has been calculate they need to be multiplied by 2.5 which will give the overall SU value which ranges from 0-100. Two averages will be calculated using the 5 scores, one from the mobile application SU values and one from system management interface SU values. These two averages will be evaluated and then combined to work out the total system average which will also be evaluated. Anything above 68 would be considered above average.

The testing environment will involve two mobile application users and one system management interface user. Due to the Yellobrick client being unavailable Hoffi employees will carry out the usability tests. Five fellow employees from Hoffi who have not been involved with this project will each test both parts of the system and complete an SU scale form shown in Figure 65 for each part of the system.

Each of completed system usability scale is available in Appendix 3.

Figure 65: System usability scale template



13.2 Test results

Here I will discuss the results from the system functional, non-functional and usability testing.

13.2.1 System functionality testing

To test the system functionality I wrote a test case for each functional requirement acceptance criteria listed in section 9.2. I then carried out these tests following the steps marking if it was a pass or fail. The results of each test case can be found in Appendix 1.

All of the tests were marked as a pass deeming the functionality of the system a success.

13.2.2 Non-functional requirement testing

To test all of the non-functional requirements excluding user-friendly I wrote a test procedure for each acceptance criteria listed in 9.3. All of the tests passed other than the compatibility requirement. The reason this failed was because the mobile application was not compatible with Android which was a desirable feature. Due to time constraints it was not possible to convert the mobile application to Android. This has now been added to future work. The full test results are available in Appendix 2.

The non-functional requirements are still deemed as a success as the Android compatibility requirement was a desirable feature.

13.2.3 Usability testing

To test the user-friendly non-functional requirement I used the System Usability Scale (SUS). I calculated the system usability scores using the SUS calculation for each mobile application and system management interface test. Each of the system usability scores for the 5 tests of each system is shown in Table 6.

Table 6: System usability score

Mobile application system usability score	System management interface system usability score
92.5	80
75	80
90	80
75	85
90	95

I then worked out an average score for each system by adding up each of the 5 scores and then dividing by 5. The average usability score for the mobile application was 84.5 and for the system management interface 84.

Using these 2 values I calculated the overall system average usability score which was 84.25. This is well above 68 that is considered above average. In the user-friendly acceptance criteria the average needed to be over 68 for the requirement to pass.

This means the usability test is deemed a pass.

13.2.4 Testing summary

Overall the system passed every mandatory requirement acceptance criteria. Therefore the system was a success.

13.2.5 Further testing

To increase confidence in the system further testing could be carried out including unit or white-box testing. Unit testing is an automated piece of code that tests an individual unit of source code, more commonly an individual function or procedure. White-box testing is a method of testing internal structures or workings of the system.

13.3 Feedback from client

Once the prototype had been finished it was presented to Alison John on behalf of Yellobrick in the environment described in 12.4. She really liked the system and the potential it offered, but needed more guarantees it wouldn't break or have lag issues before investing heavily in the technology and using the system on a large scale.

The main points from the feedback were:

- A reliable Wi-Fi connection needs to be available to all users using the interactive experience for it to function properly. When testing the application on her device at first her 3G was being used so the data wasn't being sent to the server as quickly as it should have been. This also used her own personal data which could potentially turn away customers. After she logged into the Wi-Fi the system worked as expected and this meant there were no issues with using user 3G data.
- Use the phone as a sensor rather than a screen. She really liked how the horror frame, behaviour data and monitor data delivered content. She suggested disabling the music audio from playing out of the smartphone device so the mobile application would just calculate and display how far the user is from each of the iBeacons and send data to the server.
- The application still needs to work if the technology fails. At the moment the system outputs error messages when there is a problem with the system. She wants the mobile application to work offline if there is no internet or Bluetooth technology. For this to happen the location behaviour would have to be disabled and a touch screen interactive section would need to be added to the mobile application.
- The current application doesn't notify the user when data is being pushed to the server or when the audio starts playing. She asked if it was possible to have a small popup box which appears just once to alert the user when different events are happening.
- The content was delivered to the users at the correct distance. She was pleased with how accurately and quickly the content was delivered using the users physical position.
- There is a difference in performance when using our test iPhone 5 phone compared to her iPhone 6. The iPhone 5 lagged by approximately 1-2 seconds compared to the iPhone 6 that had a delay of about half a second. She asked if it was possible to notify the user content was loading. She also mentioned about a possible notification on the home page telling users if they have an older device and warning them they may have slight lag issues.
- She discussed how there was a lot of potential with this type of technology and how Yellobrick are really interested in the story telling side of things.

- She really liked the analytical side of things where you could view the user behaviour. She discussed how this could be potentially expanded to track in more detail where users are. This would result in data being sent more often so not just on entering and exiting radii, but every few seconds or so. This would make the system more accurate and give a lot more details about the user, but it would also use more bandwidth.
- Integrate this technology with smart watches. She asked if it was possible to install this application on the up and coming Apple watches. This would require a lot more work and would probably be another project itself.
- Availability on Android. After seeing it work on iOS she asked for it to be made available on Android. This was originally a desirable feature, but after seeing the prototype she wants it included as soon as possible.

Overall Alison John on behalf of the client Yellobrick liked the system a lot, confirming all of the mandatory functionality worked. Her feedback was generally very positive and constructive. I have added the requested improvements to section 16.

14 Conclusion Phase Two: Yellobrick Prototype

Here I will summarise the system I developed in phase two.

In section 9 I discussed in detail what the Yellobrick system needed to do. Through extensive testing I achieved all of the mandatory requirements for the Yellobrick system with the only requirement to fail being the mobile application being compatible on Android which was a desirable feature.

The mobile application interacted with the iBeacons and accurately calculated the users position in real-time using the Estimote API. The users were notified through the mobile application and their data sent anonymously using a hash value instead of their UUID to a remote server when they entered and exited a specific distance of an iBeacon. The radius and zone techniques were used to update the data visible to the user and trigger an AJAX GET request so data would be sent to the database location on the server. The mobile application failed to achieve full compatibility as it only worked on iOS and not Android. Although the system could be quickly converted to Android, extensive tests would be required which were not possible due to time constraints.

The system management interface was set up on computer accessible by Yellobrick employees and allowed them to view the complete system history using the data pushed from the mobile application. This allowed them to view user behaviour anonymously. It also included other functionality such as the monitor, behaviour and horror frame pages which could be displayed to mobile application users through a large screen users to enhance their experience. The system management interface was compatible across the four major browsers Internet Explorer, Chrome, Firefox and Safari.

The mobile application and system management were both user-friendly as shown by the system usability scale results with the mobile application achieving a score of 84.5 and the system management interface achieving a score of 84, both well above the average of 68.

Overall the Yellobrick prototype was a success and achieved all of the main aims and goals as confirmed by Alison John on behalf of the client Yellobrick.

15 Project Conclusion

Here I will summarise the project as a whole referring to the original aims and objectives described in section 1.

Before starting the main investigation and implementation sections I evaluated iBeacon technology and compared it to other IPS technologies to give me an overview into the different types of IPS technologies and how they worked in 3.

The main body of the report was split into two phases. The first phase involved evaluating iBeacon operating characteristics, this helped me understand how iBeacons worked in a lot more detail.

The results from phase one were used to help with the second phase of the system which was to deliver a prototype for Yellobrick on behalf of Hoffi. The system needed to allow users to interact with the system using their physical location and Yellobrick employees to track user behaviour. This system was successfully implemented as confirmed by the client with all of the essential requirements passing their tests.

Splitting the project into two phases helped me produce a much better final system as if I skipped phase one and went straight into developing the Yellobrick system I would have been learning how to use iBeacons as I went along.

The project has altogether been a success with all of the essential aims and objectives being met.

16 Future Work

Here I will talk about enhancements for the system and things I would like to do in the future related to this project.

16.1 Make the mobile application available on Android

One of the desirable aims was to make the application available on Android in addition to iOS, but due to time restrictions this was not possible. I created a hybrid application using Cordova which allows compatibility across all the major mobile platforms including Android. To enable this application on Android I would just need to run the command 'cordova platform add android' in the project directory. This would create an Android build which could be tested on simulators and devices. The Android application would require extensive testing to ensure it runs like the iOS counterpart, as there are some minor differences.

16.2 Improve the system management interface

The original purpose of the system management interface was to allow people to view user behaviour in a user-friendly format. This meant I prioritised simplicity over security. To make the system management interface more secure a login mechanism could be installed to prevent anyone from accessing the data. This could be implemented by adding a table to the database containing usernames and passwords. On the website a login page could then replace the home page requiring a username and password. If the user enters a username and password that matches the combination in the database it would log them in successfully and display the rest of the content as it does now.

16.3 Encrypt the database

The purpose of the database stored on the server was to store users entry and exit times when entering different beacon zones. Due to the amount of data being stored in this database scalability and efficiency were prioritised over security. To increase the security of the data stored on the server data could be encrypted. Data is originally sent from the mobile application to a PHP file on the server using AJAX before being inserted into the database. It is then accessed via the system management interface using PHP. To reduce the chance of someone eavesdropping and stealing the data it could be encrypted. This could be done in the mobile application using a JavaScript encryption library such as Crypto-js (Mott, 2012) using a

secure password. When the system management interface then selects from the database it can be decrypted in PHP using the password.

16.4 Evaluate and test other iBeacons

I used Estimote iBeacons for this project due to the high level of documentation and active community. There are hundreds of other iBeacon vendors creating different types of iBeacons. A recent report evaluating sixteen different ones shows how different each of the iBeacons can be (Aislelabs, 2014).In the future I would like to test different types of iBeacons to see whether it makes a difference to performance and battery life.

16.5 Investigate the use of wearable technologies interacting with iBeacons

There is a lot of discussion about wearable technology including smart watches. Recently Apple announced they were releasing the Apple Watch which supports iBeacon technology. When the users were testing the prototype they really liked the idea of their smartphone being a sensor rather than a screen. So based on this feedback the use of smart watches would be perfect for this scenario. In the future I would like to test an app similar to my final prototype on smart watches to see if it improves the user experience and how it affects the performance of the IPS.

16.6 Combine iBeacon BLE technology with other technologies

A lot of the successful IPS combine multiple IPS technologies. In future projects I would like to develop a system which doesn't just use iBeacons, but also combines it with other promising IPS technologies such as Wi-Fi, VLC and magnetic positioning. In addition to combining these technologies I would also like to use other smartphone features including the accelerometer, gyroscope and compass. Combining all of these technologies should lead to a far more accurate IPS.

16.7 Projects for the future using iBeacon technology

Evaluating iBeacons has given me a lot of ideas for real world systems. There are the main ones like advertising specific content to users depending on their movement history and heat maps to see where the users have been, but I have also thought of a few other projects including a student attendance system where the user would sign in with their smartphone which would be tied to their student ID. This could then be further improved to open doors at university if

the user is within close proximity and has authorisation. Another interesting idea which Hoffi are very excited about are the prospect of developing interactive games based on the users physical location.

17 Reflection

This will reflect on how I managed and developed the project.

17.1 Project management

The project initially started in December 2014, when Hoffi asked me to investigate the different types of indoor positioning systems, but due to the Christmas break and exams I didn't start using iBeacons until late January. At the start of the project I created a plan which was submitted as part of the initial plan report. This plan outlined the different deliverables I would plan to do each week and when I would meet my supervisor to discuss the project.

I finished the implementation of both the prototype and system quicker than expected. I think doing this project on behalf of Hoffi helped motivate me to start the project earlier and working in an office environment helped remove all of the distractions I would have at home or in the labs. In the initial plan I said I would finish testing the different types of functionality iBeacon can offer by the end of February. In reality I finished this stage of the project mid-way through February due to working approximately 30 hours a week on this project at work. I continued with the project as planned even though I was ahead of schedule and started developing the final Yellobrick system.

The mobile application didn't take too long to develop as I referred back to the prototypes I created when I evaluated the operating characteristics of iBeacons. Setting up the database didn't take too long as I had done this many times before so I knew what I was doing. However getting the data to send from the mobile application to the remote database took a lot longer than expected. I assumed it would take a few days, but it actually took an entire week to get the data sending in the correct format efficiently. Due to finishing the IBeacon investigation quickly I was still a couple of weeks ahead of schedule so the small delay didn't matter too much. This task was due to be completed by mid March, but I had completed it by the start of March.

After successfully sending data to the database from the mobile application I started developing the web interface. This part of the system took approximately a week to develop, which is approximately the same timeframe I planned in the initial plan. This meant the system was fully implemented by the second week in March, a couple of weeks ahead of the deadline and proposed meeting with the Yellobrick client. During this two-week period I carried out

extensive tests against the functional and non-functional requirements of the system. Due to time constraints at work and other deadlines at university I didn't have time to convert the application to Android as testing each bit of functionality to ensure it worked like its iOS counterpart would have taken too long. I set aside the Easter break from the end of March to middle of April as contingency time for the project, but due to the successful start to the project I started writing my final report using notes written when I implemented the system. Once I started writing my report I realised how big my project was and I was unsure if I had enough time and words to describe each part of the system in enough detail. This motivated me to work constantly for approximately a month to finish this report.

Throughout the project I visited my supervisor countless times to discuss my project and issues that arose during the report.

Looking back at the project, if I was to start it again I would have done one thing differently. I would have written my report alongside the implementation so everything was fresh in my memory when writing about it.

17.2 Project development

The project development side of things went fairly well. I think working on this project in a work environment definitely helped me as I could talk to professionals about my project which helped improve my knowledge. The work environment also cut away distractions I would get at home and motivated me to do my project as I had the added pressure of developing a system for a paying client.

The decision of splitting the project into two phases certainly helped when it came to delivering a successful system. As I felt much more confident using iBeacon technology thanks to my investigatory work in phase one when designing the Yellobrick system.

The main issue I experienced when developing the system was sending data from the mobile application to the server which took a few days longer than expected. After carrying out a lot of research and speaking to professionals over forums I eventually got it working by adding a single line of code into the mobile applications configuration file.

Overall I think this project has helped develop my software development skills further, as I now feel much more confident with AJAX as before this project I had only used AJAX a few times previously. I also feel like my skills in HTML5, PHP and MySQLi have improved due to the use of

these languages consistently over a four-month period. I also learnt how to use new Cordova plugins and API's including the Estimote and device plugins.

This project also helped improve my communication skills, as I had to present the system to the Yellobrick client and talk to my colleagues and supervisor throughout the duration of the project. I now feel more comfortable when speaking to people about my own work.

Before this project I had never worked with positioning systems so I had very little knowledge about it and didn't even know what indoor positioning systems were. Now I feel like I know a lot about positioning systems and feel very confident when talking about them. Before starting this project I had never even heard of iBeacons and now I have created a working system that relies on them. I have gained a lot of valuable knowledge throughout this project which will help me in future projects.

There wouldn't be too much I would do differently regarding the development section if I started this again. I feel that having experience in the professional workplace where I developed multiple projects similar to this helped me choose the suitable methods when developing this project.

18 Appendices

Appendix 1: System functionality test case results.

Appendix 2: System non-functionality test results.

Appendix 3: Completed usability SU scales.

Appendix 4: Redman SQL database file.

19 References

Aerohive Networks, Inc. (2014) *Aerohive iBeacon Integration*, [Online], Available: http://www.aerohive.com/pdfs/Aerohive-Solution_Brief-iBeacon_Integration.pdf [28 Apr 2015].

Aislelabs (2014) *The Hitchhikers Guide to iBeacon Hardware: A Comprehensive Report by Aislelabs*, [Online], Available: http://www.aislelabs.com/reports/beacon-guide/ [28 Apr 2015].

Amazon (2007) *Nintendo Wii Controller (Wii)*, [Online], Available: http://www.amazon.co.uk/Nintendo-RVLACJW1-Wii-Controller/dp/B000IMWK2G [28 Apr 2015].

Anthony, S. (2012) *Think GPS is cool? IPS will blow your mind*, [Online], Available: http://www.extremetech.com/extreme/126843-think-gps-is-cool-ips-will-blow-your-mind [28 Apr 2015].

Baskerville, J. (2014) *Bluetooth Low Energy Devices Part 2: iBeacons explained*, [Online], Available: http://www.cogapp.com/blog/bluetooth-low-energy-devices-part-2-ibeacons-explained [28 Apr 2015].

Blueimp (2014) *JavaScript-MD5*, [Online], Available: https://github.com/blueimp/JavaScript-MD5 [28 Apr 2015].

Brenchley, M. (2014) *UK smartphone take-up is almost on a par with laptop ownership*, [Online], Available: http://theappgarden.co.uk/blog/business/uk-smartphone-take-almost-par-laptop-ownership/ [28 Apr 2015].

Brooke, J. (1986) *http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html*, [Online] [29 Apr 2015].

Bytelight (2014) *How it Works*, [Online], Available: http://www.bytelight.com/ [28 Apr 2015].

Cangeloso, S. (2013) *Forget WiFiSlam — ByteLight uses LEDs for indoor positioning*, [Online], Available: http://www.extremetech.com/extreme/151068-forget-wifislam-bytelight-uses-leds-for-indoor-positioning [28 Apr 2015].

Carhartl (2011) *jquery-cookie*, [Online], Available: https://github.com/carhartl/jquery-cookie [28 Apr 2015].

Community Health Maps (2014) *How Accurate is the GPS on my Smart Phone? (Part 2)*, [Online], Available: http://communityhealthmaps.nlm.nih.gov/2014/07/07/how-accurate-is-the-gps-on-my-smart-phone-part-2/ [28 Apr 2015].

Cordova (2014) *Plugin APIs*, [Online], Available: https://cordova.apache.org/docs/en/4.0.0/cordova_plugins_pluginapis.md.html#Plugin%20APIs [28 Apr 2015].

Cordova (2015) *The Command-Line Interface*, [Online], Available: https://cordova.apache.org/docs/en/4.0.0/guide_cli_index.md.html [01 Feb 2015].

Costa, T. (2013) *Indoor Venues Are The Next Frontier For Location-Based Services*, 23 Jan, [Online], Available: http://www.forbes.com/sites/forrester/2013/01/23/indoor-venues-are-the-next-frontier-for-location-based-services/ [28 Apr 2015].
Cotrell, L. (2012) *TULIP Algorithm Alternative Trilateration Method*, [Online], Available: https://confluence.slac.stanford.edu/display/IEPM/TULIP+Algorithm+Alternative+Trilateration+Method [30 Apr 2015].
Estimote (2013) *Estimote Real-world context for your apps*, [Online], Available: http://estimote.com/ [28 Apr 2015].
Estimote (2014) *Estimote Application*, [Online], Available: https://itunes.apple.com/gb/app/estimote/id686915066?mt=8 [Apr 2015].
Francica, J. (2014) *Solving Indoor Positioning - SiRFusion: CSR's New SDK*, [Online], Available: http://www.directionsmag.com/entry/solving-indoor-positioning-sirfusion-csrs-new-sdk/425594 [28 Apr 2015].
Free Space Optics (2011) *Visual Light Communications*, [Online], Available: http://www.freespaceoptics.org/visuallightcommunications.html [28 Apr 2015].
Google (2014) *Go inside with indoor maps*, [Online], Available: https://www.google.com/maps/about/partners/indoormaps/ [28 Apr 2015].
Hern, A. (2014) *What is Apple's iBeacon?*, [Online], Available: http://www.theguardian.com/technology/2014/jan/13/what-is-apple-ibeacon-retail-tracking [28 Apr 2015].
Hexagon (2014) *The Search for Sustainability.*, [Online], Available: http://geospatialworld.net/uploads/magazine/Geospatial-World-August-2014.pdf [28 Apr 2015].
Hoffi (2012) *About*, [Online], Available: http://hoffi.com/ [28 Apr 2015].
Jaakko Vuorio, S.N.B.A. (2014) *Indoor Positioning System*, [Online], Available: https://wiki.aalto.fi/display/MEX/Indoor+Positioning+System [28 Apr 2015].
kdzwinel (2014) *trilateration.js*, [Online], Available: https://gist.github.com/kdzwinel/8235348 [28 Apr 2015].
Kevin Curran, E.F.T.L.J.A.S.D.W.D.A.M. (2011) *An evaluation of indoor location determination technologies.*, [Online], Available: http://www.researchgate.net/publication/220232962_An_evaluation_of_indoor_location_determination_technologies [28 Apr 2015].
Marcacci, S. (2014) *LED Lighting Efficiency Jumps Roughly 50% Since 2012*, [Online], Available: http://cleantechnica.com/2014/11/05/led-lighting-efficiency-jumps-roughly-50-since-2012/ [28 Apr 2015].
Mott, J. (2012) *crypto-js*, [Online], Available: https://code.google.com/p/crypto-js/ [28 Apr 2015].

Nan Wu, X.L. (2011) *RFID Applications in Cyber-Physical System*, [Online], Available: http://www.intechopen.com/books/deploying-rfid-challenges-solutions-and-open-issues/rfid-applications-in-cyber-physical-system [28 Apr 2014].
Newman, J. (2010) *Apple Abolishes Wi-Fi Scanners From App Store*, [Online], Available: http://www.techhive.com/article/190789/wifi_scanners_banned.html [04 28 2015].
Nielsen, J. (2000) *Why You Only Need to Test with 5 Users*, [Online], Available: http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ [29 Apr 2015].
Owano, N. (2012) *Finland team uses Earth's magnetic field for phone indoor positioning system*, [Online], Available: http://phys.org/news/2012-07-finland-team-earth-magnetic-field.html [28 Apr 2015].
Page, C. (2015) *Apple's UK market share climbs to 42.5 percent as Android dominance falters*, [Online], Available: http://www.theinquirer.net/inquirer/news/2389029/apples-uk-market-share-climbs-to-425-percent-as-android-dominance-falters [28 Apr 2015].
Piotr Krawiec, W.B. (2015) *How to build beaconified Apple Watch apps using Estimote's SDK & Nearables Simulator*, [Online], Available: http://blog.estimote.com/post/112708221260/how-to-build-beaconified-apple-watch-apps-using [28 Apr 2015].
Pnazarino, M. (2013) *What exactly WiFiSLAM is, and why Apple acquired it*, [Online], Available: http://thenextweb.com/apple/2013/03/26/what-exactly-wifislam-is-and-why-apple-acquired-it/ [28 Apr 2015].
Puchta, O. (2014a) *What are Broadcasting Power, RSSI and Measured Power?*, [Online], Available: https://community.estimote.com/hc/en-us/articles/201636913-Broadcasting-Power-RSSI-and-Measured-Power-explained [28 Apr 2015].
Puchta, O. (2014b) *What are the characteristics of Beacons' signal?*, [Online], Available: https://community.estimote.com/hc/en-us/articles/201029223-Beacons-signal-characteristics- [28 Apr 2015].
Ratcliff, C. (2014) *What are iBeacons and why they might change marketing?*, [Online], Available: https://econsultancy.com/blog/64492-what-are-ibeacons-and-why-they-might-change-marketing/ [28 Apr 2015].
RFID Centre (2013) *Indoor RFID Tracking*, [Online], Available: http://www.rfidc.com/docs/indoor_rfid_tracking.htm [28 Apr 2015].
Robert Lilley, P.D. (2008) *Loran*, [Online], Available: http://www.loran.org/ [28 Apr 2015].
Saltzstein, R.N.B. (2012) *Bluetooth Low Energy vs. Classic Bluetooth: Choose the Best Wireless Technology For Your Application*, [Online], Available: http://www.medicalelectronicsdesign.com/article/bluetooth-low-energy-vs-classic-bluetooth-choose-best-wireless-technology-your-application [28 Apr 2015].
Santos, A. (2014a) *What are advertising and connectivity modes?*, [Online], Available: https://community.estimote.com/hc/en-us/articles/201030983-Advertising-and-Connectivity-Modes- [28 Apr 2015].

Santos, A. (2014b) *What are potential sources of wireless interference?*, [Online], Available: https://community.estimote.com/hc/en-us/articles/200794267-Potential-Sources-of-Wireless-Interference [28 Apr 2015].
Teknomo, K. (2012) *How K-Nearest Neighbor (KNN) Algorithm works?*, [Online], Available: http://people.revoledu.com/kardi/tutorial/KNN/HowTo_KNN.html [28 Apr 2015].
Thompson, D. (2013) *Guide to iBeacon Hardware*, [Online], Available: http://beekn.net/guide-to-ibeacons/ [28 Apr 2015].
Williams, P. (2014) *Relative & Absolute Location Services*, [Online], Available: http://localz.co/blog/relative-absolute-location-services-compared/ [28 Apr 2015].
Yellobrick (2014) *Who we are?*, [Online], Available: http://yellobrick.co.uk/#about-section [28 Apr 2015].
Zahid Farid, R.N.M.I. (2013) *Recent Advances in Wireless Indoor Localization Techniques and System*, [Online], Available: http://www.hindawi.com/journals/jcnc/2013/185138/ [28 Apr 2015].
Zahradnik, F. (2013) *Wi-Fi Positioning System*, [Online], Available: http://gps.about.com/od/glossary/g/wifi_position.htm [28 Apr 2015].
Ziflaj, A. (2014) *Native vs Hybrid App Development*, [Online], Available: http://www.sitepoint.com/native-vs-hybrid-app-development/ [28 Apr 2015].