

Using Beacons for Location based targeted advertisement to nearby users.



Dheeraj Thankachan
C1417381/CM3203

May 2017

One Semester Project (40 Credits)

Supervisor: Alia I Abdelmoty
Moderator: Michael Daley

Acknowledgements

This project would not have been possible without the help and support of God. I thank God for giving me strength and patience to complete my project and report.

I would like to thank my project supervisor, Dr. Alia Abdelmoty, for encouraging, supporting and providing me with excellent guidance and valuable feedback throughout the course of the project.

I would like to thank my parents for their invaluable support and encouragement. I could not have done this without them. I am also grateful to my sisters, Divya and Deepthy and my brother Deepak for providing me guidance support and encouragement.

I would also like to thank my friends, Tom Léost, Khalid Rahman, Jacob Martin, Jake Ling and Zack Ahmed for their encouraging and helping me stay motivated.

Abstract

Beacon technology is changing the way consumers interact with brands, making devices more helpful, smart and ultimately revolutionizing the way retailers measure the offline impact of online ads and engaging nearby users with information that they wish to discover and be interested in.

In this project, I will be developing a mobile & web application and the infrastructure that allows beacons to provide/transmit targeted advertisement to nearby users. This project also involves physically making equipment out of Raspberry Pi and Bluetooth modules to carry out the operation.

Table of Contents

Acknowledgements	2
Abstract	3
1. Introduction	8
1.1. Aims and Goals	8
1.2. Project importance	9
1.3. Audience and potential beneficiaries	9
1.4. Scope of the project and outcomes	10
1.5. Approach	11
1.6. Assumptions.....	13
2. Background	14
2.1. Problem Identification and Context	14
2.2. User Tracking Theory and Possible Scope	15
2.3. Constraints to the approach	16
2.4. Exploring existing solution	17
2.4.1. Estimote.....	17
2.4.2. Insiteo	17
2.5. Methods and Tools for Solution	18
2.5.1. Language choice for Web & API development	18
2.5.2. Language choice for mobile app development	19
2.5.3. Server Side Technology	19
2.5.4. Frameworks and JavaScript	19
2.5.5. Twitter Bootstrap.....	20
2.5.6. JQuery & AJAX	20
2.5.7. Data Storage	21
2.6. I Beacon Construction	22
3. Specification.....	23
3.1. Application/Program Specification	23
3.2. Essential Requirements.....	23
3.3. Desirable Functionality (Time permitting requirements that would enhance the application)	24
3.4. User Interface Specification for the mobile and web application	25

4.	System Design & Architecture Overview	26
4.1.	iBeacon overview	26
4.2.	iBeacon Construction & Setup.....	27
	Step 2: Connect to the Raspberry Pi	27
	Step 3: Wi-Fi Setup	27
4.3.	Core Tracking Algorithm & Data Propagation System	31
4.3.1.	iBeacon ranging.....	31
4.3.2.	Data propagation and aggregation System.....	32
4.4.	IOS Interaction Controller	33
4.5.	Rest API Architecture	34
4.6.	Web Application	34
4.7.	Mobile Application	34
4.8.	Business Logic	34
4.9.	Database	35
5.	Use Case Analysis	37
5.1.	Shop Owner and web application.	37
5.2.	Phone user and mobile application.	39
5.3.	Mobile phone and beacon range	40
6.	Design.....	41
6.1.	IOS Storyboard.....	41
6.2.	Web application user interface	43
6.2.1.	Inventory page	43
6.2.2.	Add new item to the inventory.....	44
6.2.3.	The active heatmap page	45
6.2.4.	Beacon page	46
6.2.5.	Add a new beacon page	47
7.	Implementation	48
7.1.	Development Process.....	48
7.1.1.	Development Process for API.....	48
7.1.2.	Development Process for Mobile Application	48
7.1.3.	Development Process for Web Application	48
7.2.	Implementation Details and Technical Challenges	49

7.3.	Implementation Process for API.....	49
7.3.1.	End Points.....	49
7.3.2.	API Usage.....	49
7.4.	Implementation Process for Mobile Application	51
7.4.1.	Triggering events when in a beacons proximal range.....	51
7.4.2.	Updating real-time location to the data base via API	54
7.4.3.	Retravel of data from API.....	54
7.5.	GUI implementation of the mobile app.....	55
7.5.1.	Homepage.	55
7.5.2.	Navigation drawer.	56
7.5.3.	Product category page.....	57
7.5.4.	Detailed product view page	57
7.5.5.	Confirmation page.	58
7.6.	Development Process for Web Application.....	58
7.6.1.	Adding a product to inventory.....	58
7.6.2.	Displaying all products from the inventory.....	60
7.6.3.	Editing a product's details in the inventory	61
7.6.4.	Deleting a product in the inventory	61
7.6.5.	Deleting a beacon from the system.....	61
7.6.6.	Displaying all beacon in system	62
7.6.7.	Adding a beacon to the system	63
7.6.8.	Editing a beacon's details in the system	64
7.7.	Displaying the active heatmap of store floor activity	65
7.7.1.	The concurrent AJAX call to update the canvas	66
8.	Testing.....	69
8.1.	Overall Approach to Testing	69
8.2.	Use Case Testing.....	69
8.3.	Functional and Interface Testing.....	70
8.4.	Automated Testing	70
8.5.	Algorithm Testing	70
8.6.	Test Result.....	73
9.	Evaluation	74

9.1.	Functionality Fulfilment	74
9.2.	Project Methodology and Management.....	74
9.3.	Technology and Design Evaluation.....	75
9.4.	Web technology.....	76
9.5.	Mobile Technology	76
9.6.	Server Side Technology	76
9.7.	Data Management.....	76
9.8.	Software Design Evaluation	77
9.9.	Hardware Evaluation	77
9.10.	Implementation Evaluation	77
9.11.	Testing Evaluation	78
9.12.	Future Development	78
9.13.	Conclusion	79
10.	Annotated Bibliography	81
11.	Appendices	82

Table of Figures

Figure 1	Feature Driven Development.	11
Figure 2	triangulation	15
Figure 3	Data propagation and aggregation System	32
Figure 4	IOS Interaction Controller	33
Figure 5	Rest API Architecture	34
Figure 6	DB Design 1	35
Figure 7	DB Design2	36
Figure 8	Use Case 1.....	37
Figure 9	Use Case 2.....	39
Figure 10	Use Case 3.....	40

1. Introduction

1.1. Aims and Goals

The overall goal of this project is to explore the use of beacon technology, in order to provide an immersing and targeted experience for shoppers, and allowing shop owners to collect meaningful real-time data from store floor activity, which can then aid them to make smarter and strategic business decisions.

In order to satisfy the needs and goals of the project, I will be will be developing an IOS mobile application that will be used by the shop customers, the back-end software and hardware infrastructure that allows beacons to transmit targeted advertisements to nearby users. In order to create the low voltage transmitting beacons, I will be using Raspberry Pi's and Bluetooth modules. The application will also have a web back-end, which will allow the shop owner to add new products to the inventory, create a new beacon fenced area and view real-time analytics of store floor activity.

A core component of the project is the proximity algorithm that is used to compute the user's location in relative to the store. This allows us to calculate how close we are to the emitting source, this information allows to interact with the physical environment and provide feedback to the user. The system will also collect this user proximal data, as this information will allow us to create a real-time visual heat map of user's travel path and current activity.

There are very few applications that allow the user to carry these tasks out the in App store, all few of these offer very basic functionality and all the technology and hardware used is proprietary to their company. I believe there is huge room for improvement in how extensible and modular, the software architecture and hardware can be built.

Furthermore, with this application and its hardware, the biggest goal is to be able to have hi extensible and modularity. Which allows the users to create and manage large ad-hoc beacon network & providing functionalities that allow developers to interact and collect user data real-time.

1.2. Project importance

The functionalities of the application & the system design, of existing products in the market isn't offering extensible development as its technology and hardware used is proprietary to their company, this resulted in the apps and hardware offering very basic functionality and limited future development. I believe there is huge room for improvement in how extensible and modular, the software architecture and hardware can be built.

The other important factor is the amount of analytical data, that one can collect in order to make smarter and strategic business decisions. Beacon technology is changing the way consumers interact with brands, making devices more helpful, smart and ultimately revolutionizing the way retailers measure the offline impact of online ads and engaging nearby users with information that they wish to discover and be interested in. From the collected information it allows us to a later date to deduce what areas are most popular and create a user travel log, which contains which path the user took and how long they stayed in a specific area for.

My preliminary research into this area suggests that there are only very few software and hardware solutions, which provide this added (or much other) functionality which performs such a service which I aim to provide. Thus I feel there is an actual need to such decentralized and modular development.

1.3. Audience and potential beneficiaries

The audience for my project assumes that they are active shoppers, who are looking for a much more immersing and targeted experience in shopping, they really want to be notified the of the latest offers and products. An assumption is made that the users have no technical abilities and have not used an immersing and targeted application before thus ensuring clean and simple design.

As my project also offers a web application for the shop owners the audience of that sector assumes that they can read analytical data in a graphical format, yet assumes no technical aptitude or even moderately complex statistical domain knowledge. My project is only available on the IOS platform for mobile and web application for the store owner and will require a working network connection. I will also only be developing the application in English for obvious reasons (see the scope below for more information).

The target audience for the system has no age restriction, however, the application will be most beneficial for shop owners, who have a large and busy store. As it will provide them with information, regarding store floor activity in a clean and intuitive manner.

The preliminary users for the mobile app are aged between 18 and 35 who are actively seeking new technology to streamline their shopping lifestyle. The general design of the mobile app and web application is going to be clean and simple so it is easy to navigate around and once learned how it operates, thus allowing them to operate it without having to think too much. The web application will have tools in a formal layout as the main purpose of the backend is to be informative as opposed entertainment since the general purpose to manage and display the collected data from the beacons.

1.4. Scope of the project and outcomes

The scope of this project limits the functionality of the features to the user in the beacon's proximal range. The core scope of the application and resulting outcomes are as follows:

- Design and develop an algorithm that computes the proximity between a transmitting beacon and mobile device.
- Design and develop a web platform that allows users set desired behavior to specific beacon when in proximal range of its user, and visualize store floor active in a heat map
- Design and implement a mobile user application that responds to a beacon's characteristics when the phone enters in the beacon's proximal range
- Collect feedback from testing in order to support the study of how to use beacons to provide Location-based targeted services to nearby users.
- Build the application and hardware in hi-extensible and modular manner.

To solve the problem of "hardware inaccessibility" I will be building the low voltage transmitting beacons using Raspberry Pi's and Bluetooth modules.

- Construction of low voltage Bluetooth beacons that transmit information to mobile devices.
- To be able to configure the beacons to emit the frequency that is required for I beacons.

iBeacon Spoofing can sometimes occur, I Beacon signals can be emitted by any BLE emitter. Current the iBeacon specification does not provide a method to authenticate the iBeacon signal, that is transmitted from the beacon. This is because Proximity UUID, Major, and Minor values are broadcasted publicly, then makes it possible for people to forge an iBeacon, by using an emitter and configuring same broadcasted values.

Realistically viewing the project scope and that time that is required to develop each of the core functionalities, a more intense focus will be on developing and designing the foundation of

each functionality, as my biggest aim of this project is to enable extensible and modular, development of software architecture and hardware development, and then adding the necessary features to improve it after completion and testing stage.

1.5. Approach

For my application's development, I've chosen to adhere to Feature Driven Development (FDD). This software development cycle best suits my needs as I have pre-defined design goals.

Feature Driven Development encourages to invested completely in one stage at any given time To design and develop features while ensuring quality work. It also enables be to be aware of the design and to change the design as needed. [10]

FDD consists of five processes during the design and development of the application and they are- develop an overall model, build a feature list, plan by features, design by feature and build by feature. [10]

Figure 1 shows the project life cycle of Feature Driven Development.

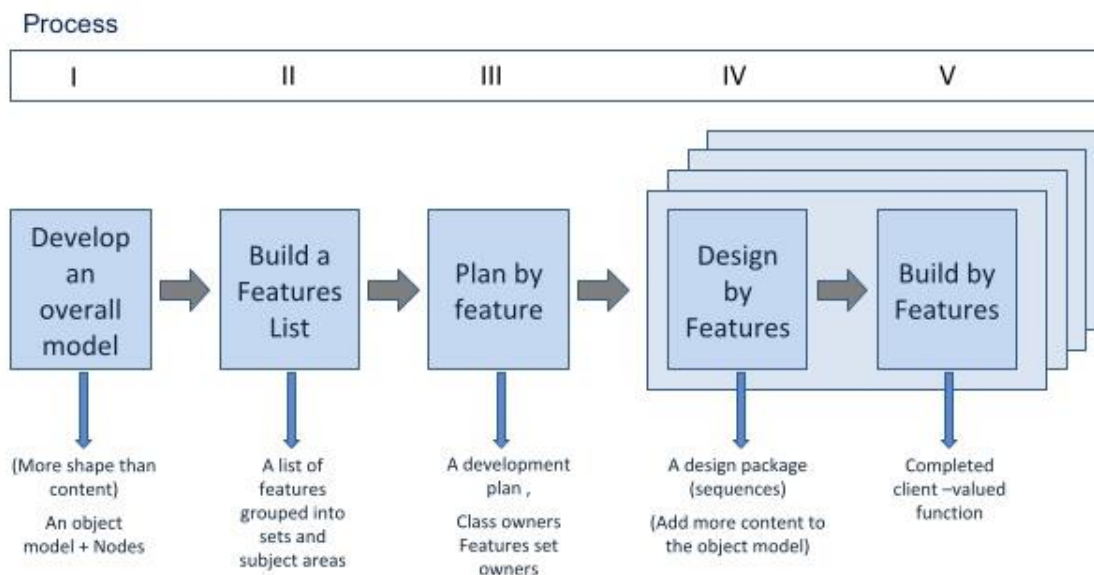


Figure 1 Feature Driven Development.

As my biggest aim of this project is to enable extensible and modular, development of software architecture and hardware development. Feature Driven Development enables me to reduce the cost of changes in requirements by having multiple short development cycles, rather than a long one. This is integral for such decentralized and modular development while ensuring quality code and modularity.

Normally the modeling team consists of domain experts, the chief architect and development members. The team domain members then perform a high-level walkthrough of the scope of the application. Since the project is an individual project it was not possible to follow all the FDD's procedures but certain aspects of FDD was applied to the project. The overall model for the essay scoring application was created based on the fundamentals of the application. A high-level walkthrough of the application was done by researching different aspects of the project and to delineate the main functionalities of the system.

This involved creating class and sequence diagrams for the application. This was then reviewed at a later stage in the development phase to check if any changes were needed at the design stage if it was required, then it was applied. The second stage involved identifying all the features for the application and creating a feature list. The functionalities needed in the mobile application and web application was then analyzed and a list of features was produced. Each feature was further broken down into smaller feature sets. This involved considering the risk factors and the complexity of implementing the features for the application. After the feature list was created, it was checked by the project supervisor to ensure the features were feasible to be completed within the time span available and within scope.

The next step involved planning the order of implementation of the features and this was based on the complexity of the features. Since there are no development teams, the development sequence was worked out and each feature to be implemented was set based on the difficulty. The next stage involved implementing the features planned from the previous stage. This also involved testing the feature developed to ensure that the implementation meets the expected functionality of the system. FDD defines six milestones to track the progress of the project. The project was tracked using this approach because it is easier to manage and monitor the work done on a daily basis and to calculate the workload. To make the tracking process easier a 'percentage complete' is assigned at each milestone. Throughout the whole project, I had been organizing meetings with my project supervisor regularly to get feedback, mentorship, and suggestions during each stage. The first three milestones of the project were done during the design by feature stage and the last three milestones were completed during the build by feature stage, after all the features were implanted, each of the features were tested against the initials goals and requirements.

1.6. Assumptions

In developing this project there are a few assumptions I adhere by.

- As a before mentioned the end-product will be a fully functional prototype so there will be more focus on meeting the functional requirements rather than having a visually appealing application.
- The project's locational scope has been limited to the users that have the app installed and active on the store floor, where it is beacon fenced.
- There is an active internet connection, via either Wi-Fi or Data, on the user's IOS smartphone. If not, the application will not be able to push or pull the data from the API.
- There is an active Bluetooth connection, on the user's smartphone. In order to interact with the beacons. If not, the application will not be able to detect the user.
- There is an active internet connection, on the shop owner's computer to access the Web application back end. If not, the application will not be able to push or pull the data from the API.
- There is an active internet connection, on the shop owner's computer to access the web back end. If not, the application will not be able to push or pull the data from the API.
- Assume user's phone is running is IOS 9 or above. This is based on a recommendation to not program to anything before this version due to security vulnerabilities. [1]

2. Background

2.1. Problem Identification and Context

Beacon technology is changing the way consumers interact with brands, making devices more helpful, smart and ultimately revolutionizing the way retailers measure the offline impact of online ads, and engaging nearby users with information that they wish to discover and be interested in.

Nearby notifications make shopping better by helping shoppers get things done with their phone, such as scanning items to get reviews or splitting the bill for their table in a restaurant. The technology also makes it easier for consumers to get useful information, whether that means seeing if an item is in stock or locating that same item while in-store.

While these kinds of communications often rely on other types of location signals, beacon signal stake precision and quality to the next level, compared to other technologies such as geofencing. For its users, beacons can give a brand an edge in a world where Omni-channel shopping is becoming the norm and consumers expect to get what they want when they want it. A survey conducted by Food and Drug Administration(FDA) shows that shoppers are more willing to shop at places where they are able to locate a specific item in less than 2 minutes and wish to learn more about they buy, this shows us that having the information I fan item is in stock or locating that same item while in-store and providing in-depth information about and item is vital for customers

Shop owners and companies are now incorporating such technology, as it enables them to collect analytical data, which can be used at a later date to deduce what areas are most popular, what are the peak times and create a user travel log, which contains which path the user took and how long they stayed in a specific area for. Having this vital information, it enables them to make, smarter and strategic business decisions.

The industry leader for beacon related research is Eddystone by Google, Eddystone is a protocol specification that defines a Bluetooth low energy (BLE) message format for proximity beacon messages. Many other competitors have released similar products such as estimate, where technology and hardware used is proprietary to their company and charges for any usage on their platform. But the core problem is, most of these providers lack opens and flexible in software and hardware architecture prohibiting the development.

This project aims to develop and improve how extensible and modular, the software architecture and hardware can be built.

2.2. User Tracking Theory and Possible Scope

Whilst doing my initial research into potential methods & technology for the project, there were a couple user tracking algorithms that I discovered that enables me to locate the user from a beacon and their distance from the emitting source, this was essential it would serve as fundamental basis of what tasks and items should be displayed once the user enters a proximal range. [20]

The other method that I have explored makes use of triangulation. Triangulation method uses signal and strength readings from multiple emitting sources to position the user in respect of the tire-fenced area, it is then used to calculate the distance between the broadcasting and receiving devices. For this method to work, three or more reference devices must be positioned in a way they cover an area. Next, distances to the reference devices are roughly calculated by the receiving device and an interception point between these is found, which is usually where the user is located. The simplified concept of this method is shown in Figure 2 below.[20]

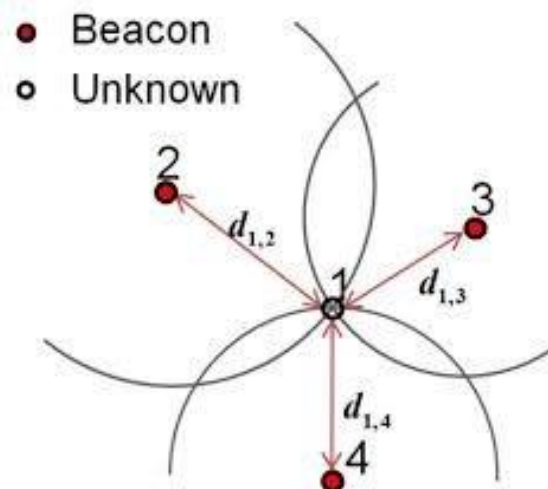


Figure 2 triangulation

This method seems very feasible but requires a lot more computational and more research needs to be conducted to reduce the interference and square error. [21]

2.3. Constraints to the approach

One of the biggest constraint to project is the time that is required to develop each of the core functionalities, as this project is highly ambition & complex, if a functionality of takes longer than the expected time, it could affect the project drastically. Partnered with the fact I'm learning iOS Programming, if I did not fully grasp how to create a mobile app, I will not be able to develop anything.

The other factor is, I have a lack of access to hardware and software tools when developing & testing the software. For the entirety of the project, I am the sole person who has invested in buying all the hardware and software that is required to carry out this project.

The application also requires the user to have an internet connection to be able to retrieve results. The application also needs have Bluetooth enabled and location services to be active when accessing location, to be able to provide some of the intended features for the application.

There could be a possibility of iBeacon signal spoofing or interference can occur, I Beacon signals can be emitted by any BLE emitter. Current the iBeacon specification does not provide a method to authenticate the iBeacon signal, that is transmitted from the beacon. This is because Proximity UUID, Major, and Minor values are broadcasted publicly, then makes it possible for people to forge an iBeacon, by using and emitter and configuring same broadcasted values.

2.4. Exploring existing solution

There are several third-party companies that tried to provide the beacon technology and hardware infrastructure. Upon first inspection of the two most popular of these applications, it was immediately clear that they were either lacking features or were too complex to use. The two applications I examined were 'Estimote' and 'Insiteo'. Both applications take your current location in retentive proximity of the transmitting beacon to, display the relevant information, in the targeted region. A further in-depth analysis of how both solutions operated is described below.

2.4.1. Estimote

Estimotes is an iBeacon system and it licensed iBeacon Technology from apple. The system consists of Bluetooth smart-chips wrapped in waterproof soft silicon casing, making it suitable to workable in harsh conditions. Estimote provides the developer with an application programming interface (API) which developers can use to easily develop their own tailored applications and use cases.

The nodes have the same properties as described for iBeacons and are mainly marketed towards retail solutions. The vendor does state that high precision localization is not the primary goal. Instead, the range areas defined for iBeacon are used to give an estimate location of where in a building a user is located given that the beacon position is known.

2.4.2. Insiteo

Insiteo, is French based company which has been working to create indoor localization technology. The Insiteo has several interesting technology approaches in development, where the newest concept is an indoor-GNSS (Global navigation satellite system) with hardware that re-transmits GPS and Galileo signals indoor. The new system requires almost no setup and calibration work while providing high precision accuracy better than 1 meter. The system will also work without these GNSS repeaters. RF-signals such as Wi-Fi and Bluetooth smart are fused together with internal sensors such as accelerometer, compass, and gyro in a smartphone to provide indoor positioning with high precision and responsiveness. The system supports multi-storey buildings and Insiteo claims to achieve an accuracy of 2 meters or better. Insiteo also combines the positioning service with several other business oriented use cases such as geofencing, location aware push messages and visitor monitoring.

2.5. Methods and Tools for Solution

To create the system, I will be using a number of selected methods, technology's, design decisions and hardware to construct a fully operational beacon node and IOS app that responds to the beacon's transmitted data. The justification for the selected methods and technology for the project as follows.

2.5.1. Language choice for Web & API development

It was decided that Object-oriented (OOP) programming approach programming was the best choice for this project compared. Using OOP would allow me to approach the problem in a Model-View-Controller (MVC) fashion. In MVC, the program is built from the objects of three types and they are models, views, and controllers. The model would represent the logic of the program, view defines what is presented to the user and they are normally HTML, JavaScript or AJAX that contains variables passed from the controller. The controller handles the model and controls which view should be displayed according to the user requests.

Ruby on Rails (ROR) & Django was one of the first choices that were considered since it had a mature framework and most of the libraries were open source. Also, the framework provided Create-Read-Update-Delete (CRUD) functionality compared to PHP. Rails also had a great emphasis on REST-full application design and had a good testing framework. Even though there were several advantages for the language, the performance of Rails applications is not as fast as PHP. The other factor is, my experience on using Ruby on Rails was very limited compared to PHP, as I have built fully functional web apps using PHP.

The learning curve could also compromise the overall development process if I decided to develop in ROR, so realistically viewing the project scope and that time that is required to develop each of the core functionalities for the web app and API, it was not a good language choice for the project in comparison with PHP.

The web application and the REST API was decided to be written using PHP, JavaScript, CSS and HTML5. The main reason for choosing PHP was that it is the most common language for web development. From my research, I found out that PHP is built to work well with the web and can easily be integrated with MySQL, as MySQL was my selected choice for the database. PHP is also highly compatible with web servers and many operating systems compared to other scripting languages which make the framework more compatible. It is made explicitly for server-side programming and it operates faster compared to other scripting languages because it does not use much of the system's resources to run. PHP contains a wide variety of libraries and tools which were beneficial for this project.

Another advantage of choosing PHP was that it provided extensive support for database and it was also possible to embed PHP code with HTML which made it flexible compared to other scripting languages. This is very beneficial for rapid prototyping when building the web

application. Most of the client-side interaction code was decided to be written in JavaScript to provide a better user-friendly interface and lower the load on the server.

2.5.2. Language choice for mobile app development

One of my main tasks was to develop the iPhone app, as I have never developed an iPhone app before, I took on this challenge by taking programming courses in iOS development in Swift on Udacity, that was made by industry leaders from Google and Airbnb. After completion of the online course, it enabled me to become familiar with almost everything I needed to develop an app using Swift. Taking these courses further enabled me to build apps and empowered me to further design and develop my project app, with much better functionalities and user experience. [7]

Apple provides an official IDE “XCode” with all the extensive functionality and hardware accessibility. The IDE provides a vast amount of resources for automatic generation of necessary XML files and inbuilt libraries to help with designing the UI. Furthermore, it provides an emulator on which you can test your application as well as providing support for managing all imports/external APIs Hooks.

2.5.3. Server-Side Technology

Many service side technologies were explored for the project and the decision of choosing the right web server was based on the cost, support and the capabilities of the server provider (AWS). Hosting my own PC as the web server & API Server was not a suitable decision for the project due to storage limitations, security threats and the risk of downtime. These conditions were vital when building a fault tolerant highly available system architecture.

One of the main benefits of choosing dedicated server is that it is more reliable compared to using PC as the server. There is a high risk of the web pages being slow and even possibly crashing as the traffic for the website increases, whereas when hosting the app with AWS, they use load balancers to spread the payload across several machines. [2]

The API and web application was decided to be hosted on a dedicated server which was running on Linux. Two open source server environment were carefully considered to run PHP, Nginx and Apache. Nginx uses fewer memory resources and can load balance connections to the server, thus, decreasing the latency. As I will be using many concurrent requests I will be using Nginx to serve my PHP files. PHP, MySQL, and Nginx can also easily be deployed using LEMP (Linux, Nginx, MySQL, and PHP) stack on Ubuntu 16.0.

2.5.4. Frameworks and JavaScript

The idea of a responsive web design is to provide a single HTML document to all the devices and this can be done by adding various style sheets based on the screen size of the device in order to add an optimized layout for the device. Having a responsive web design solves the

issues of the evolving device diversity in order to make the code work on multiple screen resolutions.

The use of JavaScript is growing in creating more exciting websites and this is where the front-end development frameworks are becoming popular in creating complex applications. After analyzing different framework choices helped make a better decision for creating a responsive web design. It is quite tricky to implement a responsive web design that can work on all the devices and hence, the framework can help in boosting the efficiency and performance of the web application. Using a framework makes the web application highly flexible to satisfy the needs for different web browsers and tablets etc.

2.5.5. Twitter Bootstrap

Bootstrap is an open source framework built by Twitter specially made for making the websites responsive and it is one of the most widely used framework for web development since it contains powerful web design tools for creating consistent cross-browser applications. It is a single CSS file which contains the layout and style support and proved to be helpful for front-end development of the framework.

I decided to use Bootstrap for the project to make the site responsive and for easier front-end development of the web application. Using the framework allows me to create the user interface of the website with user interface and experience in mind. So using bootstrap for the web app front-end development enabled me to create the app responsive design thus, making it compatible with different platforms. Compared to using other frameworks like ionic, AngularJS or Skeleton, Bootstrap was highly documented from the community which meant that it was possible to modify the code and optimize it even further, with very little time. [3]

2.5.6. JQuery & AJAX

JQuery is a JavaScript library which has a set of cross-browser methods that allows for the selection DOM elements, making AJAX requests, traversing the DOM along with many other functions. JQuery makes it easier to control HTML events, animations, event handling and other interactions on the web page. Another advantage is that JQuery elements will still be displayed even though JavaScript is disabled on the webpage when compared with other applications like Flash, this would be essential real time data. Also, it would not make the load time of the web pages slow compared to Flash technology. Since it was possible to have great looking effects on the website, using the JQuery library, it was a better choice when compared to other JavaScript libraries like shadow box or Ext JS.

AJAX (Asynchronous JavaScript and XML) is the method of exchanging data with a server, and updating parts of a web page, without reloading the entire page. AJAX is necessary for the development since the web application has an interactive heat map and users requests to be processed immediately. Asynchronous communication calls to the web server can be done with AJAX and this reduces the waiting for all the data to arrive in the client browser. AJAX

enables the pages to update instantly when an action is performed so that the interaction between the web application and the user is quicker.

AJAX is much faster compared to using an HTTP request on the page since they take a while to load. The server does not have to convert the data into HTML mark-up because the data can be sent straight to the client side. The information inside the browser will be processed by the client side code therefore the HTML display can be updated dynamically. This is because a full post back is eliminated and thus the utilization of network is reduced so the operations are much faster. This reduces the server overheads and hence the performance and speed of the web application will be faster. The combination of PHP with AJAX it will be enable me to build real time updating features such as heat map.

2.5.7. Data Storage

When selecting an open source relational database management system, PostgreSQL, MySQL, NoSQL and SQLite were taken into consideration.

The decision of choosing the best for this project was based on their speed, storage requirements, and non-functional storage requirements. Compared to the other database management system SQLite does not have a user management system and lacks protection against data loss, and is not meant for highly concurrent access. Also, it was slow in operation since it only lets one single operation to take place at any given time and so it allows a restricted amount. For these reasons, I did not select SQLite for my project.

As I have selected PHP to use for the server side language for the Rest API and the web application, NoSQL was one choice that was considered however it also had some major drawbacks compared to the others. As one of my requirements is provided real-time analytics of store floor activities, NoSQL such as MongoDB is not acid compliant, there for the data is not consistent sometimes up on access. NoSQL puts performance and scalability first; consistency isn't really a consideration". As my project is prototype and stability is not my core priority, NoSQL was not an appropriate choice for this project.

After carefully considering all the factors I plan to use MySQL, a relational database management system that is open source. As DB server will have to process many requests concurrently and in turn access the database concurrently, MySQL facilitates these requirements. The main advantage of using MySQL to me is that that it can process out responses very quickly because it uses multi-threading. Also, it has been tested for its robustness and stability. As I previously have experience of using MySQL and setting up MySQL server, I found the syntax to be easy to remember which will make the initial server-database connection easy to set up and configure.

2.6. I Beacon Construction

In order to create the iBeacon, I will be using Raspberry Pi 2 & BLE enabled Bluetooth USB adapter. I have purchased these items from Adafruit. I have followed the instruction to create from “makeuseof.com” well-known hardware development community.

Hardware I will be using includes:

- A Raspberry Pi 3
- Power supply for the Raspberry Pi
- SD card for the raspberry pi
- BLE enabled Bluetooth USB adapter.

BLE is designed specifically for transmitting over short distances, but with lower power requirements. Small packets of data, known as “advertisements” (not in the marketing sense), are broadcast by the beacon and used to trigger actions in smartphone apps, which can then be processed to display a commercial message or prompt the reader to perform an action. The main reason why I have decided construct my own beacons is because it will allow me to dynamically program and set triggers, this offers greater hardware accessibility than purchasing a company one. A further in-depth analysis of how the beacon is constructed can be seen in the System Design & Architecture Overview section. [9]

3. Specification

3.1. Application/Program Specification

The outcome of this project is to have a fully functional iOS and web application that provides most of the features, in improved and modular manor. The web application is going to allow the shop owner to add items to beacon fenced area, and from the mobile application perspective when a user enters a beacon fenced area it will display the items accordingly. The application also aims to provide analytics to the shop owner about real time instore activity.

Given below are a list of essential and desirable requirements for the project

3.2. Essential Requirements

1. The application must be able to locate and calculate the distance of the nearest beacons.
 - This method relies on the assumption the user is on the app somewhere within a reasonable distance of a nearby beacon's proximity range if not no proximity driven events will be triggered.
2. The application should be able to interact with nearby beacons and display selected notifications/product information when the user is in defined proximal range.
 - This relies on the assumption the user is within a reasonable distance of a nearby beacon's proximity range if not no proximity driven events will be triggered.
 - The method relies on the assumption, if there is an edge case when multiple signal/distance from a beacon are the same, we display all their information in an appropriate manner.
3. The software and hardware infrastructure will be created in a modular fashion thus providing greater extensibility for the project in a later date.
4. Each of the beacon nodes should work a fault tolerant structure.
5. Easy to Use Interface, the mobile application's user interface will need to be as intuitive.
6. Create a web platform that can be used to program each of the beacon nodes and its desired interaction & mange the shop settings.
 - The desired interaction can be in the form of nearby notifications or screen changes with relevant information.
 - In the web platform, the shop owner will be able to add and delete items in his inventory, these added items are the products that are displayed in the mobile application.

Justification:

Requirement 1,2 are vital to the for the project as it addresses the core technology that is used to carry out the user tracking in the project, the functionality on 2 enable users to react to the emitted data from beacon range, for this project this project to succeed these two requirements must be met.

Requirement 3, 4: These are required as it enables me to create a product works in a modular fashion and fault tolerant, these requirements enable to create the system with greater extensibility for a later date.

Requirement 5: As this project is very complex and demanding, making the user interface intuitive is essential, as too much information will overwhelm and confuse the user.

Requirement 6: This functionality is very important as it the only way the shop owner will be able to add and delete items in his inventory, these added items are the products that are displayed in the mobile application. This is also how he will be able to set the desired interaction that will allow him to show nearby notifications or screen changes with relevant information. This enables the abstraction of him from editing the code, thus making the mobile application dynamic.

3.3. Desirable Functionality (Time permitting requirements that would enhance the application)

- To get the application to collect detailed user proximal data and log interaction times.

Justification:

From the collected detailed user information, it will allow us to perform data mining in a later date to deduce what areas are most popular and create a user travel log, which contains which path the user took and how long they stayed in a specific area for. This can be hugely beneficial analytical data that the shop owner can use, to perform strategic marketing campaigns and promotions.

3.4. User Interface Specification for the mobile and web application

1. The Main Menu/Home will be accessible from every section of the application.
2. Keep all phrases and textboxes as brief as possible without sacrificing the meaning of the text.
3. Material design standards will be used when designing the front end of the web and mobile app.
4. The layout and color scheme will be kept formal and consistent throughout the application.
5. Any system messages and error messages will be displayed and handled gracefully.
6. The System status will be visible to the user at all time by having the title a top of the page depending on the active section the user is currently on.

Justification:

Requirement 1: This ensures that the user will be able to navigate to the menu, where all main functions will accessible, to ensure easy navigation.

Requirement 2: This comes from a recommendation from the IOS developer documentation, that states 'people are more likely to skip pages if the text within a page is too long & crowded', there for when developing the screens, I will ensure, not to crowd the canvas with too much text.

Requirement 3: Material design standards aims to develop a single underlying system that allows for a unified experience across platforms and device sizes. These standards are set by Google and used by developers across the world, by using these design standards it will enable me to create a natural experience for the users, the main aim is use familiar components to what a user has seen in the past.

Requirement 4 & 6: The background and layout of the form will remain consistent and be similar to all the other forms in the program; I will be using a light gradient for the background so that the text can easy be readable when placed on top. The positioning of logo and title of the form will also remain consistent as this gives a professional outlook to the program.

Requirement 5: Any system messages and error messages have handled gracefully, else it lead to bad user experience, and may even lead to users not using the application.

*Note: I will generally try my best to adhere to all the 'good' Nielsen heuristic principles as referenced to try and produce a good-quality application. [5]

4. System Design & Architecture Overview

The following sections outline the various aspects of the system architecture & design which shows the structure of the implementation and the construction of the hardware that was used carry out the project.

4.1. iBeacon overview

iBeacon is designed specifically for transmitting over short distances, but with lower power requirements. Small packets of data, known as “advertisements” (not in the marketing sense), are broadcast by the beacon and used to trigger actions in smartphone apps, which can then be processed to display a commercial message or prompt the reader to perform an action. The main reason why I have decided construct my own beacons is because it will allow me to dynamically program and set triggers, this offers greater hardware accessibility than purchasing a company one.

iBeacon is Apple implementation for an indoor positioning system, which provides proximity location service based on RSSI signal strength calculation. Apple has not revealed how the calculation is done though. Apple provides the developers with a simple-to-use API to integrate iBeacon facility into their applications. Any BLE emitter can emit iBeacon signal. The iBeacon signal contains an address, which identifies its emitter. The address components and common usage of each component are described in below.

Component	Description	Common Usage
Proximity UUID	A 128-bit string	To identify a company
Major	16-bit unsigned integer value, ranging from 0 to 65536	To identify building (branch) in the company
Minor	16-bit unsigned integer value, ranging from 0 to 65536	To identify are a region in the building. For example, a room in a building might be identified using three regions. It depends on the room’s size and the level of accuracy needed.

In addition, to address information, the Apple API provides the developer with another two important information about the emitted signal. This information is:

- **RSSI:** The Received Signal Strength Indicator of the iBeacon emitter.
- **Proximity:** This indicates the proximity of the device from the beacon. The device can be in one of four proximities: Immediate, Near, Far and Unknown.

The iBeacon itself does not provide a complete internal position system. It helps developers to build their own system, which is based on iBeacon. To build an internal position system in a building using iBeacon, the building must have iBeacon emitters installed in the rooms. Then use the iPhone application that is designed to detect the Beacon signals, and interpret what actions it should perform. As the location information is different in each region the application will need to need to query the backend server, which holds the respective information that is relevant to that specific location information.

4.2. iBeacon Construction & Setup

In order to build my I beacon, I will be using Raspberry Pi 2 & BLE enabled Bluetooth USB adapter. I have purchased these items from Adafruit. [9]
Hardware I will be using includes:

- A Raspberry Pi 3
- Power supply for the Raspberry Pi
- SD card for the raspberry pi
- BLE enabled Bluetooth USB adapter

Step 1: Download an Operating System Image

There are numerous OS images I could have used with the Raspberry Pi. I chose to use the Raspbian Debian Wheezy image, as it had all the latest drivers.

Step 2: Connect to the Raspberry Pi

In order to connect to the Pi, like to use the Console Cable to connect directly from my Mac but you can also connect with Ethernet & SSH or connect a monitor and keyboard.

Step 3: Wi-Fi Setup

I connected the Raspberry Pi to the internet via Wi-Fi so that it doesn't require Ethernet or a console cable for me to connect via SSH. When you have a lot of devices this is particularly important. I use the Miniature WifFi (802.11b/g/n) Module as my Wi-fi adapter.

Step 4: Install Required Libraries

There are several open source libraries and tools used in order to send the iBeacon data from the Raspberry Pi. Install the following libraries.

```
sudo apt-get install libusb-dev
sudo apt-get install libdbus-1-dev
sudo apt-get install libglib2.0-dev --fix-missing
sudo apt-get install libudev-dev
sudo apt-get install libical-dev
sudo apt-get install libreadline-dev
```

Step 5: Download, Make, & Install BlueZ Library

The BlueZ libraries provide support for the core Bluetooth layers and protocols. We must download, make, and then install the libraries. Run the following commands sequentially.

```
sudo mkdir bluez
cd bluez
sudo wget www.kernel.org/pub/linux/bluetooth/bluez-5.18.tar.gz
sudo gunzip bluez-5.18.tar.gz
sudo tar xvf bluez-5.18.tar
cd bluez-5.18
sudo ./configure --disable-systemd
sudo make
sudo make install
sudo shutdown -r now
```

The `sudo make` command can take quite a long time to complete. So allow it to take some time run. Once they have run we have successfully setup the Raspberry Pi to act as an iBeacon transmitter.

Step 6: Setting up the Raspberry Pi as an iBeacon Transmitter

We need to start transmitting a signal from the raspberry Pi, so that it can detected by the mobile applications.

As described in the overview the beacon broadcast has three important identifiers.

- proximityUUID: a unique UUID that distinguishes your iBeacons from other iBeacons.
- major: used to group related sets of iBeacons.
- minor: used to identify a iBeacon within a group.

In order to generate a unique proximityUUID I run the following command.

```
python -c 'import sys,uuid;sys.stdout.write(uuid.uuid4().hex)'|pbcopy && pbpaste && echo
```

This will generate the proximityUUID such as 5s3fsf8f64914bee95f7d8cc64a863b5 as the output. This is a 128-bit UUID that will be used in the mobile app to look for this specific iBeacon.

In order to check the raspberry pi is running, run `tools/hciconfig` and confirm it's up and running. Your output should look like this.

```
hci0:   Type: BR/EDR   Bus: USB
        BD Address: 00:1A:7D:DA:71:13   ACL MTU: 310:10   SCO MTU: 64:8
        UP RUNNING
        RX bytes:1136 acl:0 sco:0 events:61 errors:0
        TX bytes:855 acl:0 sco:0 commands:61 errors:0
```

I then used `hcitool` to configure the transmission of the UUID. Additionally, for this initial device, I will set the major and minor values as 0 by attaching 00 00 00 00 to the end of our UUID. In the end of the string I will add the RSSI value of C8.

The command to do this is, this will create the emitting packet.

```
sudo tools/hcitool -i hci0 cmd 0x08 0x0008 1E 02 01 1A 1A FF 4C 00 02 15 63 6F
```

The emitting break down packet as follows.

```
02      # Number of bytes that follow in first AD structure
01      # Flags AD type
1A      # Flags value 0x1A = 000011010
        bit 0 (OFF) LE Limited Discoverable Mode
        bit 1 (ON)  LE General Discoverable Mode
        bit 2 (OFF) BR/EDR Not Supported
        bit 3 (ON)  Simultaneous LE and BR/EDR to Same Device Capable (controller)
        bit 4 (ON)  Simultaneous LE and BR/EDR to Same Device Capable (Host)
1A      # Number of bytes that follow in second (and last) AD structure
```

Define vendor specific values.

```
FF      # Manufacturer specific data AD type
4C 00   # Company identifier code (0x004C == Apple)
02      # Byte 0 of iBeacon advertisement indicator
15      # Byte 1 of iBeacon advertisement indicator
```

Our specific UUID values.

```
FF      # Manufacturer specific data AD type
4C 00   # Company identifier code (0x004C == Apple)
02      # Byte 0 of iBeacon advertisement indicator
15      # Byte 1 of iBeacon advertisement indicator
```

Step 7: Detect Your iBeacon Signal to ensure it is working

In order to ensure the i beacon is transmitting the signal as intended to, I used an iOS app that is in the app store called '**Locate iBeacon**' that scans for any emitting beacon. Inside this app I need to provide the proximityUUID, major and minor information so it can scan for the specific beacon.

The image displays three sequential screenshots of the 'Locate iBeacon' app interface on an iOS device.

First Screenshot (Configuration): Shows the 'When transmitting, any blank values default to zero, except power, which defaults to -59.' section. The 'Name' field contains 'rp-01'. The 'Proximity UUID' field contains '636f3f8f-6491-4bee-95f7-d8cc64a863b5' with a 'Generate' button. Below are 'Major' (0), 'Minor' (0), and 'Power' (empty) input fields. A 'Transmitter' toggle is turned on. A 'Save' button is at the bottom.

Second Screenshot (Visible Beacons): Shows the 'Visible iBeacons - tap row for more info' section. It lists one beacon with the UUID '636F3F8F-6491-4BEE-95F7-D8CC64A863B5' and 'Major: 0, Minor: 0, Distance: 0.23m'. Below is a section 'Tap here to configure visible iBeacon UUIDs.' with multiple empty rows.

Third Screenshot (Beacon Details): Shows the details for the detected beacon. It features a lightning bolt icon in a circle. The UUID '636F3F8F-6491-4BEE-95F7-D8CC64A863B5' is displayed, along with 'Major: 0 Minor: 0'. Below this, it shows 'RSSI: -48 Accuracy: 0.36 Proximity: near'. At the bottom are two buttons: 'Distance' and 'Calibrate'.

As soon as I enter the information, the app was able to detect my i beacon, and provided what proximity region it was in.

4.3. Core Tracking Algorithm & Data Propagation System

The core concept of the user tracking is based on distances between an unknown position of the user and several known points of the beacons, the app makes use of this transmitted data from the beacon to query the backend server, which holds the location information, to calculate where the user is placed.

4.3.1. iBeacon ranging

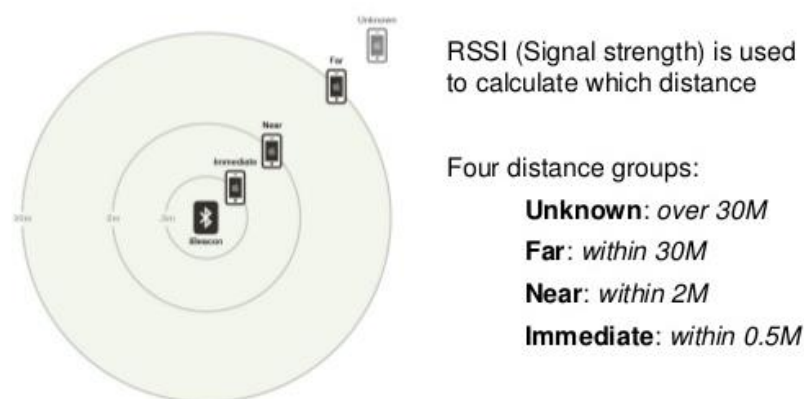
The application can request notifications when a device enters or leaves a region defined by a beacon. When an application makes this request to begin monitoring a beacon region it must specify the UUID of the iBeacon advertisement. While an app is limited to 20 regions being monitored, by using a single UUID in multiple locations, a device can easily monitor many physical locations simultaneously. Using a retail store example, a device can monitor 3 separate physical Locations (food section, clothing section, and groceries section) using the same UUID. [6]

So, when given such a distance to a known point, the geometric locus of the unknown position, the so-called Line of Position (LOP) is a circle with the known point as its center and the distance as the radius. There for now using this information, we can predict the region where the user is active at.

The categorize the beacons into four *proximity zones*:

- *immediate* (strong signal; usually up to a few centimeters)
- *near* (medium signal; usually up to a few meters)
- *far* (weak signal; more than a few meters)
- *unknown* ("hard to say", usually when the signal is very, very weak)

An illustration of how the proximity looks in respect to the distance can be seen below.



Specific events can be triggered when a phone enters each proximity zone. In our case, we will be querying the backed to get the associated data to that specific region.

4.3.2. Data propagation and aggregation System

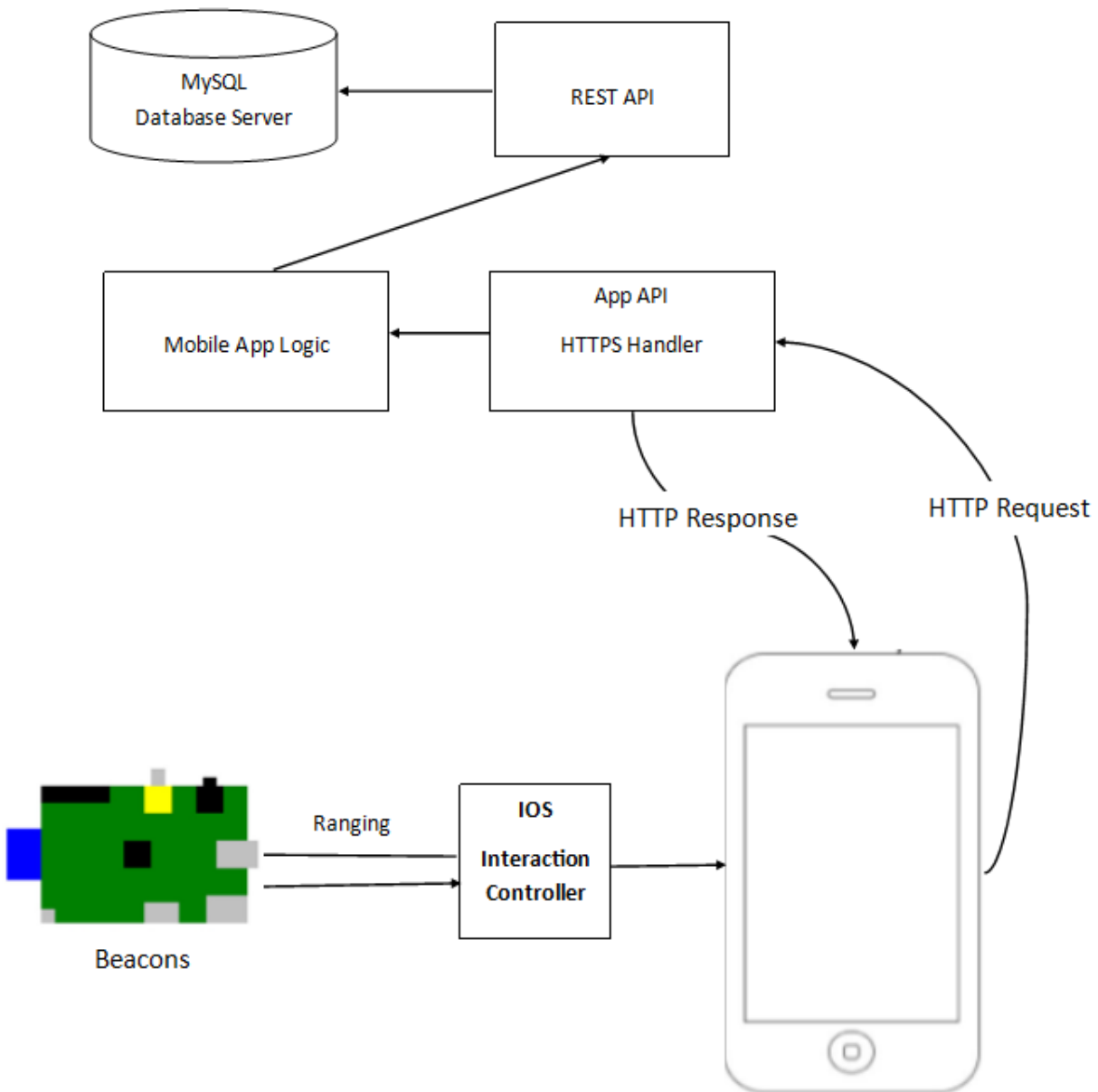


Figure 3 Data propagation and aggregation System

This is an overview of how the system works together with the whole architecture. From the data that is emitted from the beacon, the interaction controller takes this data, then passes it to API to query the backend server, which holds the location information, then this information is used to calculate where the user is in store and what data is associated with that specific location. This data is then sent back to the phone, to display the relevant content specific to that region and proximity of the beacon.

4.4. IOS Interaction Controller

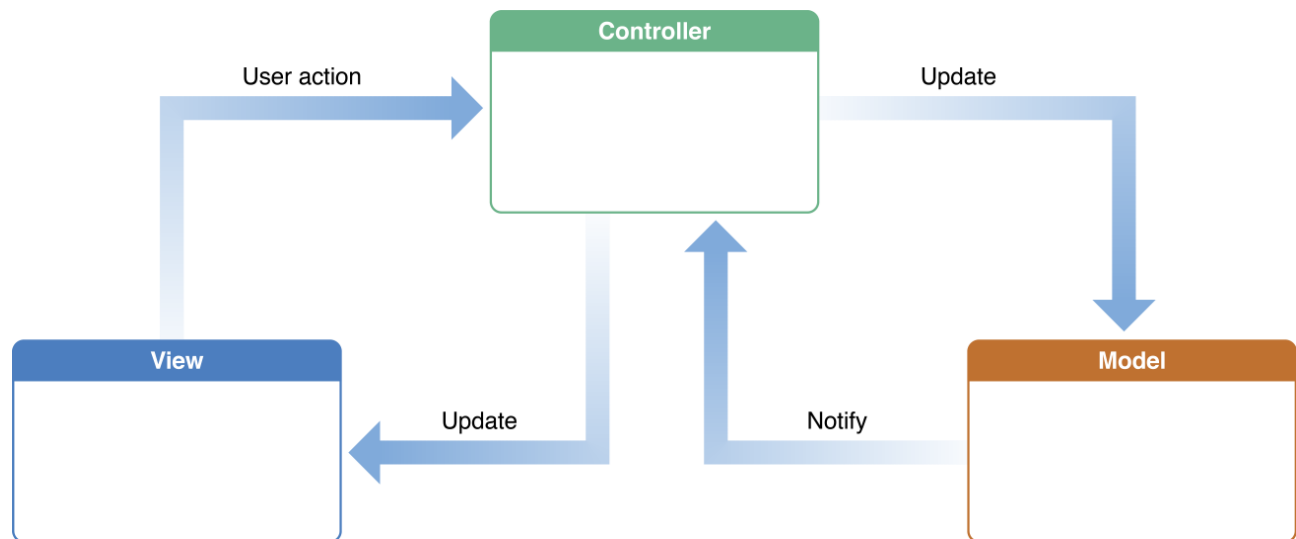


Figure 4 IOS Interaction Controller

The interaction controller handles all interactions needed to support file preview and menu display. The controller decides what to do when a change in event occurs, this event can be in the form of a user moving from beacon zone to another, a change in the proximity region or a push notification sent from the Rest-AP. This data is then set to the model, the model then looks at the new data and decides what it should do, if an interaction should be made to the user, model notifies that request to the controller, the controller then updates this information on the view.

Similarly, the user is also able to trigger an interaction, this can be in the form of moving from one app page to another, selecting a product. This user action is sent to the controller, then the controller pushes the update that is relevant to the view.

Whenever there is an active interaction with the controller, that requires data from the data base it makes use of the rest API to power all its functionalities such as retrieval of all the in-app information, processing orders, updating users position. The user tracking and beacon's proximal interaction activity relies on the API to provide with real-time information from the DB to propagate and push to the mobile and web application.

When an interaction is triggered because the user has moved to a different beacon zone or entered a different proximal range, this data is updated via the rest API. As all these events are pushed real-time to the database, this enables us to create a real-time heatmap of which beacon zone the user is at and what proximal region the user is at.

4.5. Rest API Architecture

This is the API architecture overview and how each of services and resources makes use of the end points.

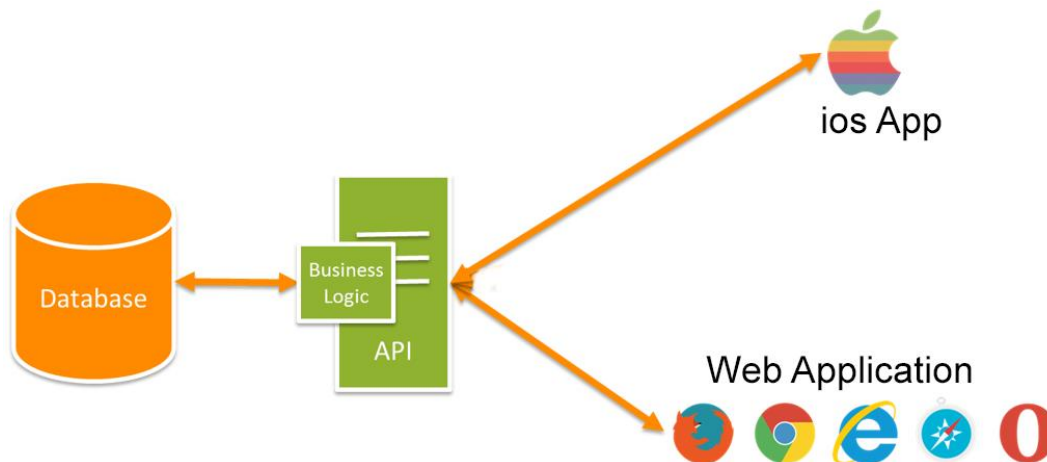


Figure 5Rest API Architecture

4.6. Web Application

The web application makes use of the API to power all its core functionalities such as insertion, reading, deletion and editing. The entire the data flow and security of the use-cases is controlled via the API. The web application relies on the API to provide with real-time information of the store activity concurrently, to visualize it as a heat map on the front end.

4.7. Mobile Application

The mobile application makes use of the API to power several of its functionalities such as retrieval of all the in-app information, processing orders. The core user tracking and beacon's proximal interaction activity relies on the API to provide with real-time information from the DB to propagate and push to the web application.

4.8. Business Logic

The Business logic is the brain that describes the operations, definitions, and constraints that apply to an activity. The operations collectively form a process; every business uses these processes to form systems that get things done. The "business rules" are inside of a "business logic" of an application is the defined parameters of the program itself such as how to handle input. For example, when we request for all the item that is present in zone 2, the logic contains

the SQL query to do this, it performs this task and returns the correct results. This allows us perform and process tasks that is higher level than (Create, Read, Update, Delete) CURD itself.

4.9. Database

The database was designed with considers the required attributes and the structure of the tables. The diagram below shows the required tables in the database and their attributes. The tables contain an auto increment ID which generates a unique identity for new rows which also helps in creating relationships between the tables. The details of each table are recorded below.

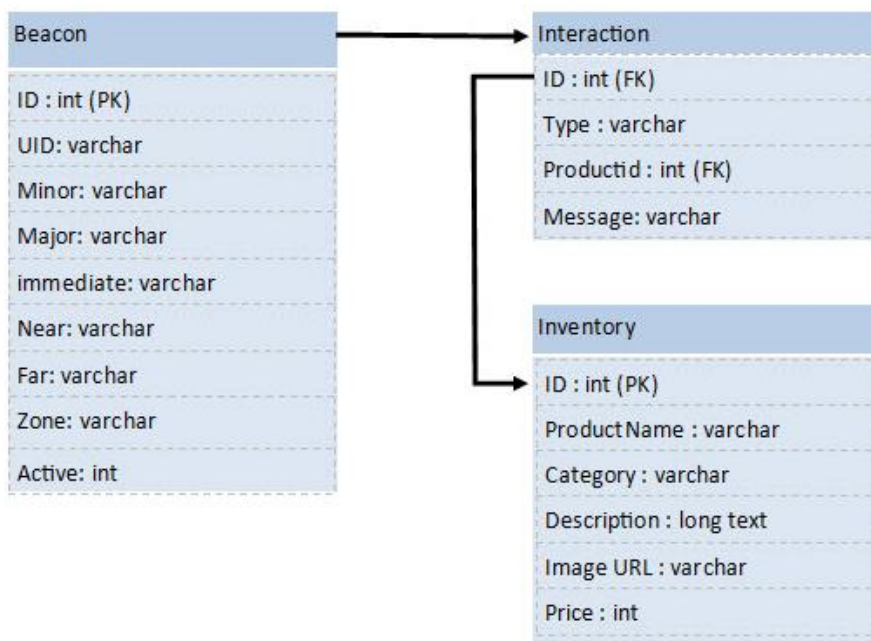


Figure 6 DB Design 1

The 'beacon' table stores all the required information about the specific beacon that is added to the system by the user such as the UUID, Major, Minor etc. It also stores the what interaction it should perform when it enters the specific proximal range such as such immediate, near, far. which is used for triggering the specific interaction once it enters the specific proximal range. The 'ID' is the primary key and it is a unique ID for each essay. This table has a 'one to many' relationships with the interactions table since each essay can have more than one interactions in different proximal range.

The table 'interaction' keeps record of the on the specific interaction that a beacon has in a proximal range and what interaction it should trigger. The field 'ID' keeps track of which beacon it is specific to; this is a foreign key from the inserted Beacon record. This table has a 'one to

one' relationship with the inventory table since it can only display/trigger one event/product in a specific proximal range.

The 'inventory' table stores all the required information about the specific item that is added to the shop inventory by the user such as the product name, category, price. The 'ID' is the primary key and it is a unique ID for each item. Data from this table is called by the API to populate the products in the Mobile and web application.

UserActivity
User ID : int (PK)
ActiveZone: varchar
Time: Date-Time

Figure 7 DB Design2

The 'userActivity' table stores all the required information about a specific user and their presence in proximal range. Data in this table is constantly updated when a user move from one region and its respective proximal range. The data is also called concurrently by the API to populate the user's activity in heat map of the web application. The 'ID' is the primary key and it is a unique ID for each active user.

5. Use Case Analysis

5.1. Shop Owner and web application.

The diagram below shows the interaction of the shop owner can have with the web application.

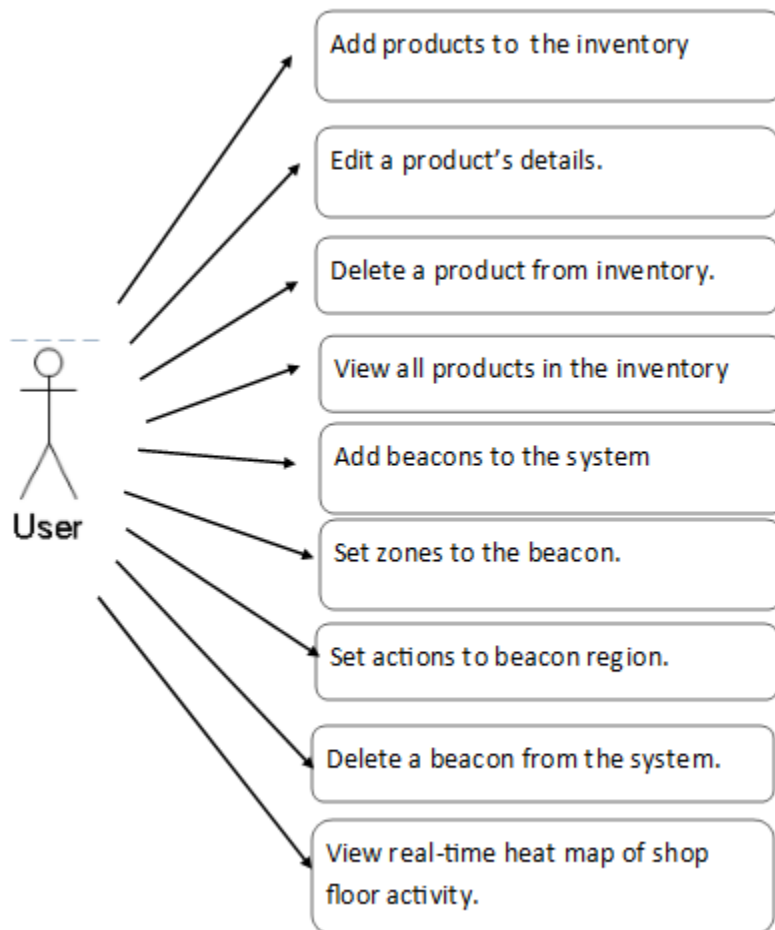


Figure 8 Use Case 1

Description of Actions

Add product to the inventory: The shop owner can, add a product with all its required fields into the system via product add form, that is accessible via the web application.

Edit a product's details: If the shop owner wishes to edit a product's information, he can do this by selecting the desired product to edit from the inventory page, then this product will opened via product editing form.

Delete a product from inventory: If the shop owner wishes to delete a product from his

inventory, he can do this by selecting the desired product to delete from the inventory, from the inventory page and then click the delete button, this will result in removing all product information.

View all products in the inventory: The shop owner can view all the products in the inventory, via the product inventory page.

Add beacons to the system: The shop owner can, add a beacon with all its required fields into the system via beacon add form, that is accessible via the web application.

Set zones to the beacon: The shop owner can classify each beacon's zone, in their respective area, via the beacon editing form.

Set actions to beacon region: If the shop owner wishes assign an interaction property to a beacon, he can do this by selecting the desired beacon from the beacons page, then this beacon will open via beacon interaction form, from here he can select the desired interaction, the beacon will do when it enters a proximal range.

Delete a beacon from the system: If the shop owner wishes to delete a beacon from system, he can do this by selecting the desired beacon to delete from the beacons page, then clicking on the delete button, this will result in removing all system information regarding that beacon.

5.2. Phone user and mobile application.

The diagram below shows the interaction of the phone user can have with the mobile application.

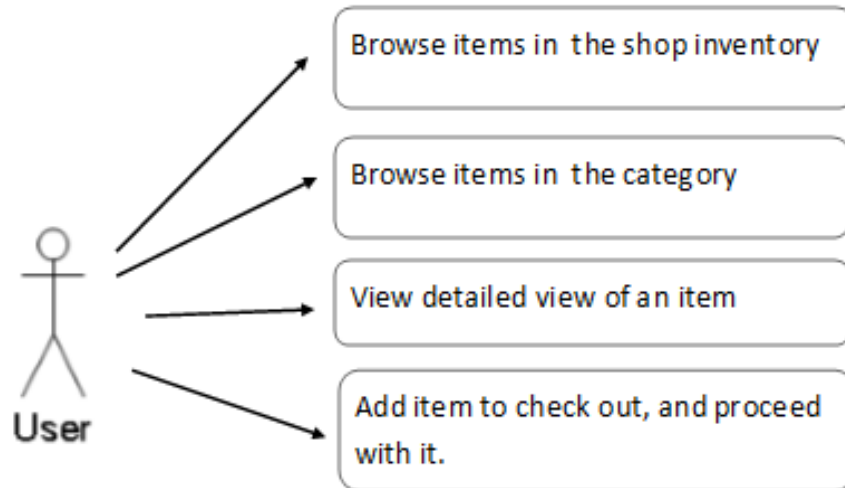


Figure 9 Use Case 2

Description of Actions

Browse items in the shop inventory: The user can view all the products that is in the shop inventory.

Browse items in the category: The user can view all the products in a specific category that is in the inventory, the category's is accessible via the home screen of the mobile app.

View detailed view of an item: The user can view a product's information in detail once has clicked on the thumbnail of the product.

Add item to check out, and proceed with it: From the product information page the user can add an item to cart and proceed to check out.

5.3. Mobile phone and beacon range

The diagram below shows the interaction of the mobile phone can have with targeted beacon fenced area.

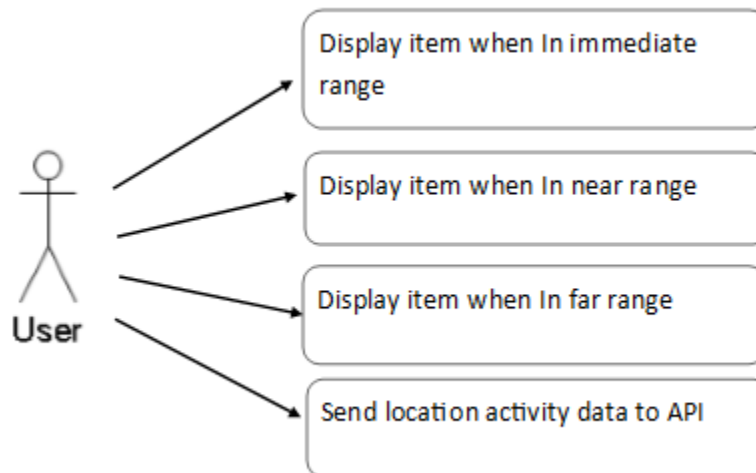


Figure 10 Use Case 3

Description of Actions

Display item when in the immediate range: When the user enters an immediate range on the beacon, the desired interaction that has been set for this specific beacon's proximal range will be triggered.

Display item when in near range: When the user enters the near range on the beacon, the desired interaction that has been set for this specific beacon's proximal range will be triggered.

Display item when in the far range: When the user enters the far range on the beacon, the desired interaction that has been set for this specific beacon's proximal range will be triggered.

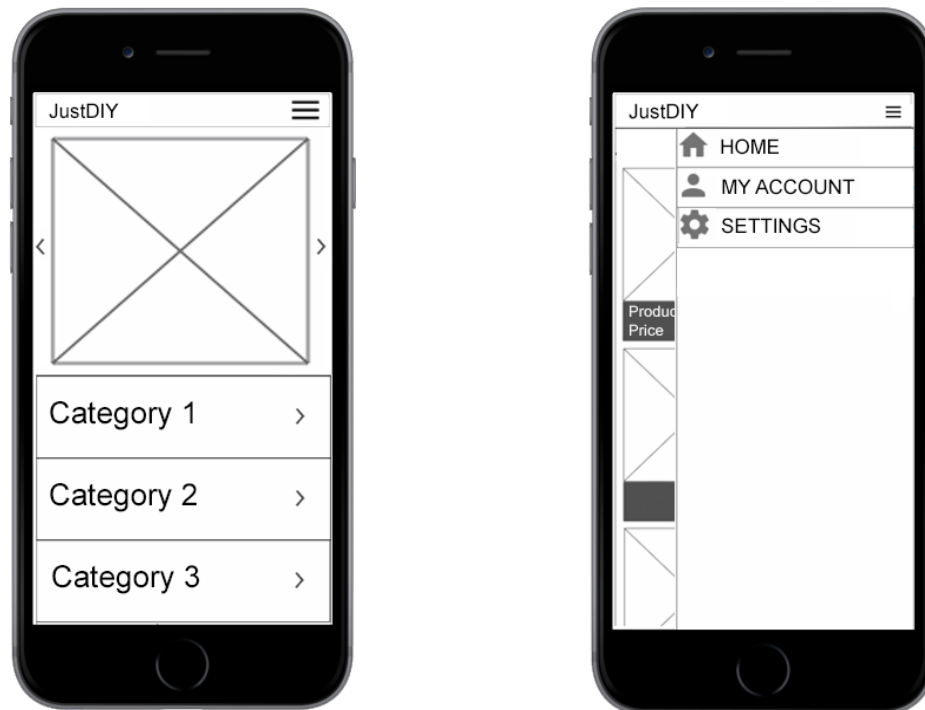
Send location activity data to API: Whenever a user enters and moves from one beacon zone to another or moves to a different proximal range, this information is updated on the database a concurrent API call in the background.

6. Design

One of the aspects of the project is the interface design for the mobile and web application. But as I before mentioned before the aim is to create the end-product will be a fully functional prototype so there will be more focus on meeting the functional requirements rather than having a visually appealing application.

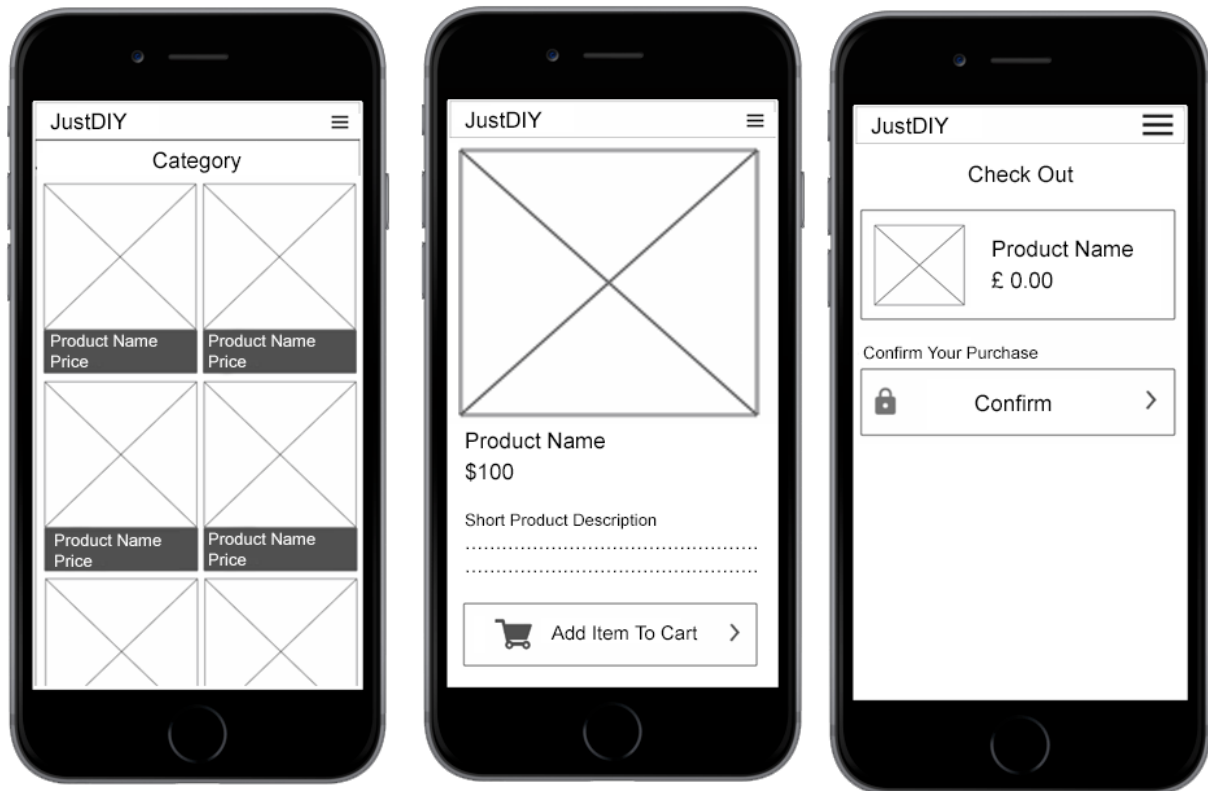
The but I still took the time to create a visually appealing and indurative application front-end that meets my user interface specification. I tried my best to adhere to all the 'good' Nielsen heuristic principles as referenced to try and produce a good-quality application. He describes usability as "a qualitative attribute relating to how easy something is to use". Nielsen's five elements on usability which are learnability, efficiency, memorability, errors and satisfaction were carefully considered while designing the interface. A speedy, responsive system is vital for not hindering good usability from being achieved. The two types of users who will interact with the system are the shop users and the shop owners.

6.1. IOS Storyboard



The first mockup is the home screen that the user is shown once the user opens the app, from here the user can transition to the specific category or open the navigation drawer by clicking on the navigation icon on the uppermost right. The mockup 2 displays what occurs when the

user press that icon, this navigation drawer allows the user to navigate to the in-app pages, and return to the home at any time



The first mockup is the category section that the user is shown once the user selected a category in the app, from here the user can transition to view a specific product by clicking on it, this will result in opening that item product detailed view (mockup 2). Each of the product displayed will be in blocks where it contains its cover thumbnail image followed by its title and price.

The second mockup is the detailed product view page section that the user is shown once the user has selected a product from the category, from here the user can add this item to the cart by clicking on 'add item to cart button', this will result in adding that item to cart and opening it cart view screen (mockup 3). The product displayed on this screen will have the enlarged image of the product followed by its title, price, and more product information.

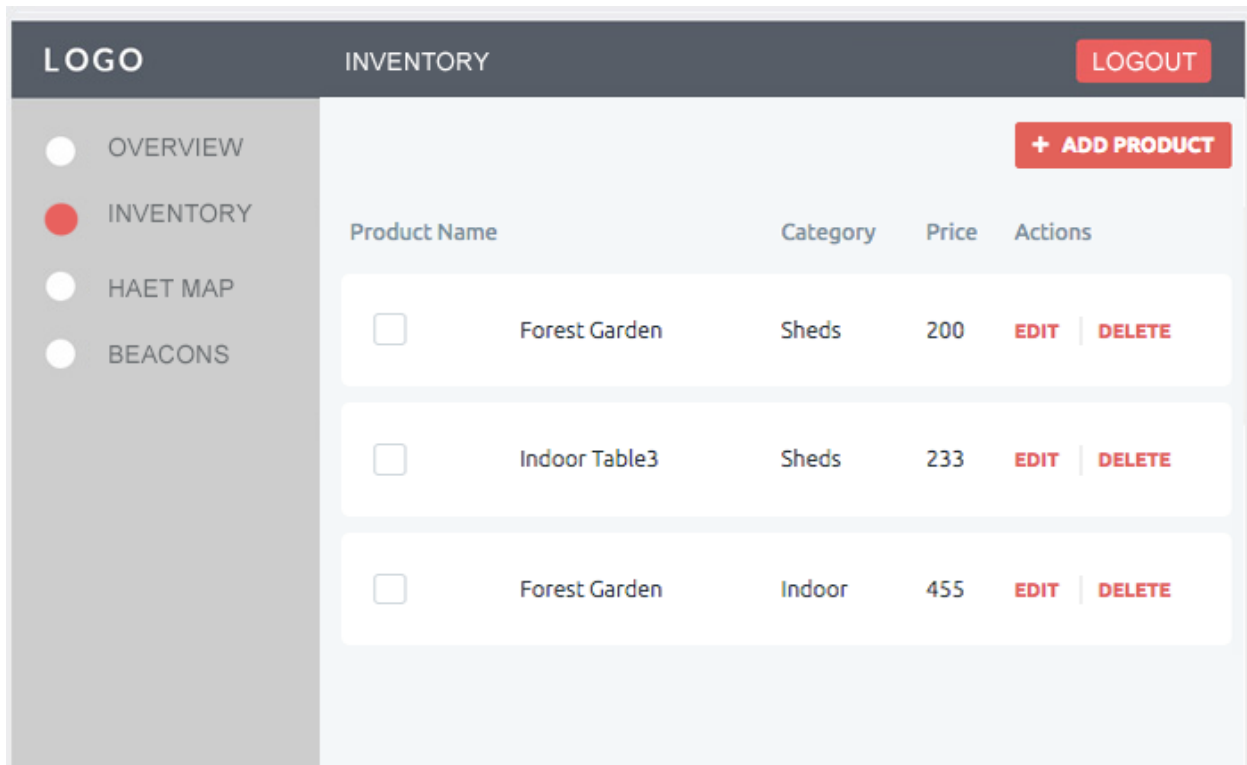
The last mockup is the checkout screen that the user is shown once the user has pressed the 'add an item to cart button'. The product that was added to the cart is displayed in blocks where it contains its cover thumbnail image followed by its title and price.

All of these screens can be triggered once the user has entered a specific beacon zone or a beacons proximity range, the interaction trigger is programmed in order to emulate a virtual store.

6.2. Web application user interface

The web application is a backend management system that allows the shop owner to manage the store inventory and beacons. The web application is designed in such way so that it can inform the user about the actions being carried on the web application. This is achieved through loading screens, input forms, the use of dialog boxes which informs the user if the action was successful or if any errors happened during the process.

6.2.1. Inventory page



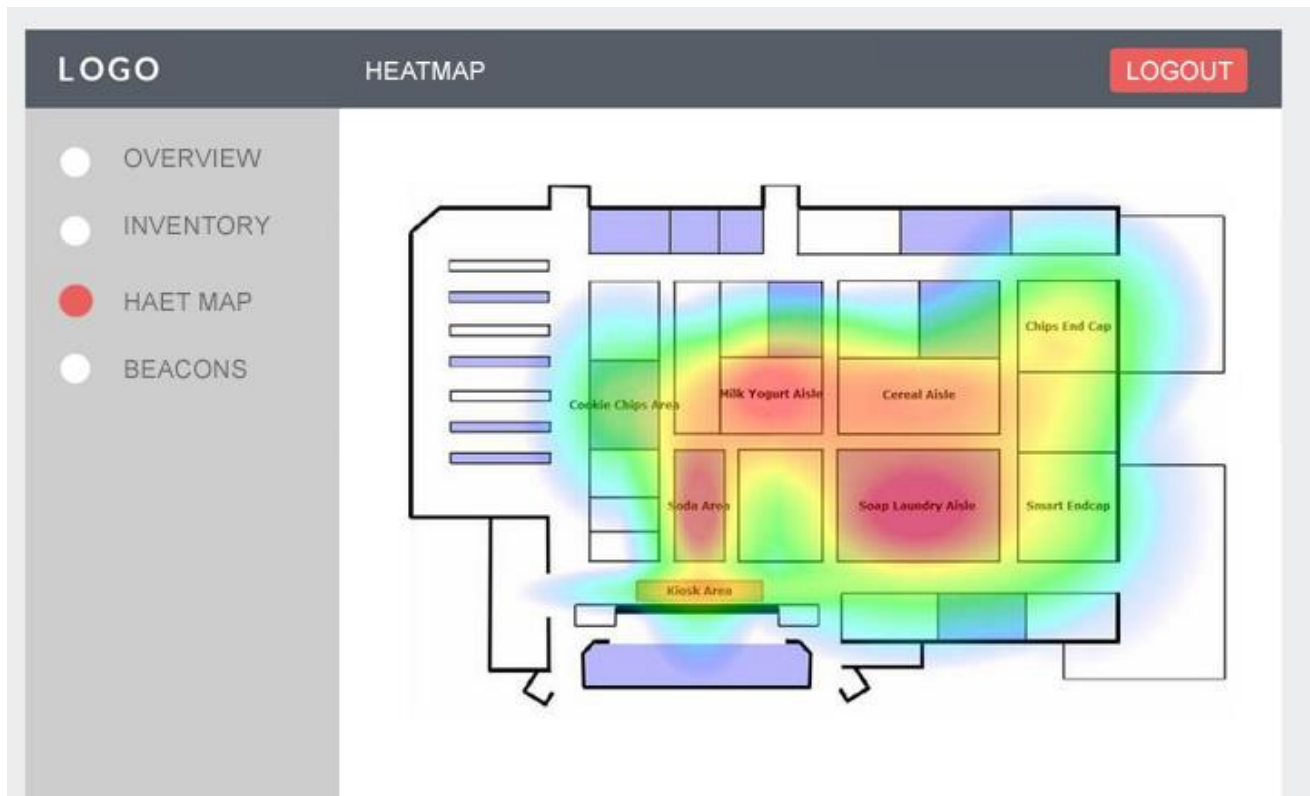
The inventory page is used to display all the items that are currently store's inventory. From this page the user can add a new item by pressing the "add product" button, this will result in opening the add a new item page. From the inventory page, the user is also able to edit the existing item and delete the item, by pressing the edit and delete button respectively.

6.2.2. Add new item to the inventory

The screenshot shows a web application interface for adding a new item to an inventory. The top navigation bar is dark grey with a 'LOGO' placeholder, the text 'Add a new item', and a red 'LOGOUT' button. A left sidebar contains a menu with four items: 'OVERVIEW' (white circle), 'INVENTORY' (white circle), 'HAET MAP' (white circle), and 'BEACONS' (red circle). The main content area has a red button labeled '< BACK TO INVENTORY' at the top right. The form itself is titled 'Add a new item' and includes the instruction 'Please fill the fields below in order to list your item.' The form fields are: 'Product Name' (text input with placeholder 'My product name....'), 'Product URL' (text input with placeholder 'Product URL'), 'Price' (text input with placeholder 'Price'), 'Category' (dropdown menu with 'Sheds' selected), and 'Product description' (text area with placeholder 'Description'). A green 'ADD ITEM' button is located at the bottom of the form.

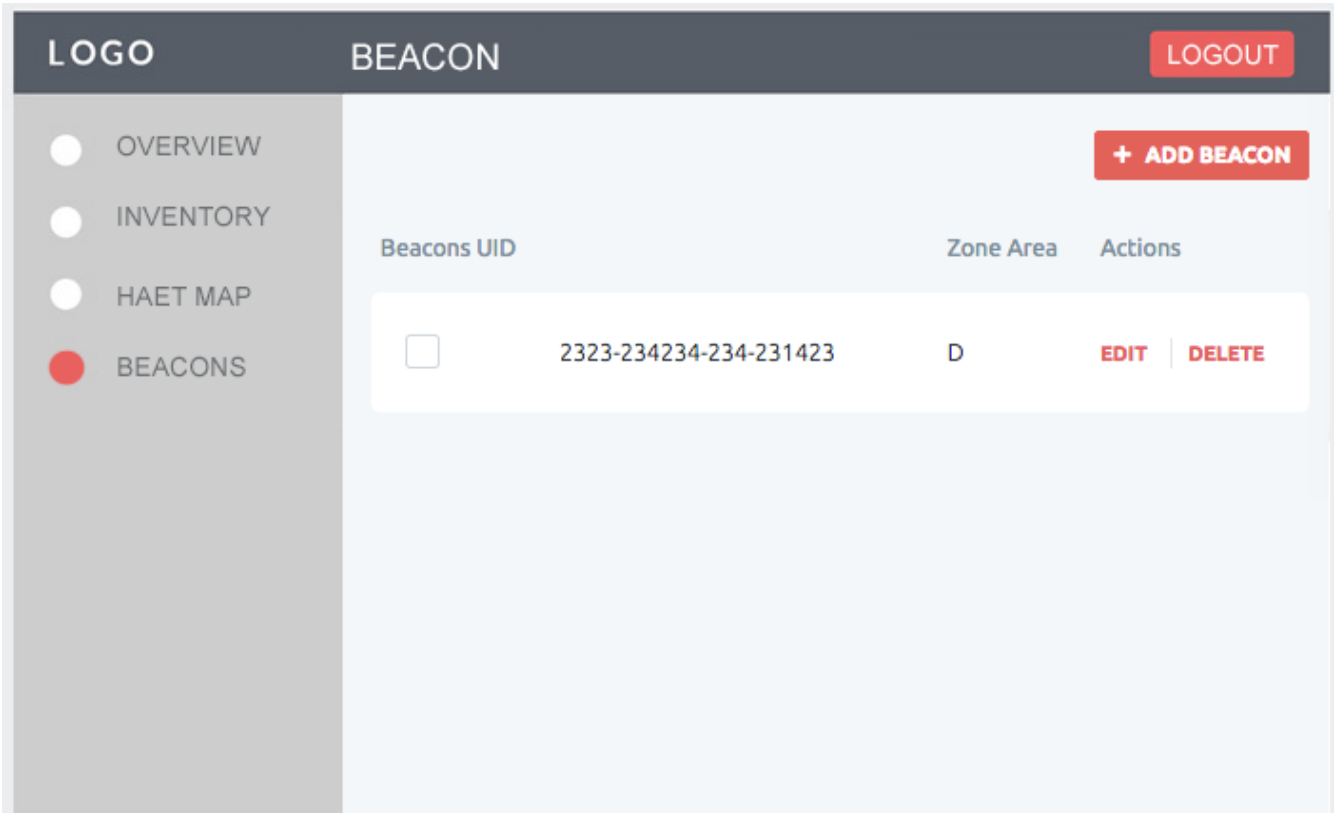
The inventory page is used to add a new item to the inventory, once the user has filled in all the required information and pressed the add item button. This data is then inserted in to the database and updated on the front end. The same form is also used for the editing purpose, so when a user selects an item to edit from the inventory page this page is opened, and all the items relevant details are populated, once saved it will be updated on the database and front end.

6.2.3. The active heatmap page



The heat map page is used to display the active users in the store in a visual representation by using a heat map. This page will make a concurrent request every 5 seconds so that it will display the real-time data on the page. The heap map will show which zone and what activity is a specific user is doing real time. The congestion of a zone is displayed by shades of colors ranging from red to blue.

6.2.4. Beacon page



The beacon page is used to display all the beacons that are currently added to the system. From this page the user can add new beacon by pressing the “add beacon” button, this will result in opening the add a new beacon page. From the beacon page, the user is also able to edit the existing item and delete the item, by pressing the edit and delete button respectively.

6.2.5. Add a new beacon page

The screenshot shows a web application interface for adding a new beacon. At the top, there is a dark header bar with a 'LOGO' placeholder, the text 'Add a new item', and a red 'LOGOUT' button. On the left, a sidebar contains four menu items: 'OVERVIEW', 'INVENTORY', 'HAET MAP', and 'BEACONS'. The 'BEACONS' item is highlighted with a red circle. The main content area has a light blue background. At the top right of this area is a red button labeled '< BACK TO BEACON'. Below this is a white box titled 'Add A New Beacon' with the instruction 'Please fill the fields below in order activate a new beacon'. Inside this box, there are two input fields: 'Beacon UID' and 'Zone Area'. The 'Zone Area' field has a dropdown arrow on its right side. At the bottom of the white box is a green button labeled 'CREATE BEACON'.

The add a new beacon page is used to add a new beacon to the system, once the user has filled in all the required information and pressed the add item button. This data is then inserted into the database and updated on the front end. The same form is also used for the editing purpose, so when a user selects a beacon to edit from the beacons page this page is opened, and all the items relevant details are populated, once saved it will be updated on the database and front end.

5.2 Design Decisions

Accessibility principles were considered while designing the interface to ensure that the web application has equal access to all users. Shneiderman emphasizes attention on consistency in his first rule for interface design. He suggested that consistency should be maintained in selecting colors, font types and naming prompts. These factors were considered and hence the layouts on pages are made consistent throughout the pages to prevent confusion. It also allows the user to adapt quickly to the application. The font sizes and colors for the marking page were also considered to make sure it was visibly clear, easy to read and legible on screen. Efficiency was another factor considered while designing the interface.

The pages on the web allocation are linked accordingly so that the shop owner can easily navigate through the pages.

7. Implementation

This chapter will discuss the different stages of the implementation process of the design & architecture planned in the previous chapters. The chapter also explains the problems encountered during the development stage and any changes that were felt necessary in the design. The design stage and implementation of the project overlapped because of the amount of spike work carried out at the design stage to reduce the amount of work needed to solve the software issues. The implementation process is aimed to explain the approaches that I took in order to achieve a specific software implementation.

7.1. Development Process

The following order of implementation was followed to ensure that the functional requirements were met with a good architecture for the code. Interface design was constantly worked to ensure the functionalities of the web and mobile application were easy to use by the users.

7.1.1. Development Process for API

1. Configuring server and setting up database
2. CURD Logic (Create, Update Read, Delete)
3. End Points

7.1.2. Development Process for Mobile Application

4. Displaying all products from the inventory
5. Displaying the detailed view of a product
6. Triggering events when in beacons proximal range.
7. Updating real-time location to the data base via API

7.1.3. Development Process for Web Application

8. Adding a product to inventory
9. Displaying all products from the inventory
10. Editing a product's details in the inventory
11. Deleting a product in the inventory
12. Displaying all the beacon in the system
13. Adding a beacon to the system
14. Editing a beacon's details in the system
15. Deleting a beacon from the system
16. Displaying the active heat map of store floor activity.

7.2. Implementation Details and Technical Challenges

The following sections will aim to explain the technical details of different functionalities in the system and the challenges faced while implementing.

7.3. Implementation Process for API

Application program interface (API) is a set of routines, protocols, and tools for building software applications. The API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components. The API makes it easier to develop a program by providing all the building blocks and creating end points which allow us to push and pull data.

The API and web application was decided to be hosted on a dedicated server which was running on Linux. Two open source server environment were carefully considered to run PHP, NginX and Apache. NginX uses fewer memory resources and can load balance connections to the server, thus, decreasing the latency. As I will be using many concurrent requests I will be using NginX to serve my PHP files. PHP, MySQL and NginX can also easily be deployed using LEMP (Linux, Nginx, MySQL, and PHP) stack on Ubuntu 16.0.

7.3.1. End Points

POST, GET, PUT and DELETE correspond to create, read, update and delete and relate to basic database operations. API endpoints describe available operations on data exposed by the service. How an end point operates is by having urls that provide data (GET request) or an url where you can submit data to (POST request). Many data formats could be used but the more common are JSON and XML, for my application the selected data type is JSON. GET `/api/ShopItems/123456` will return data on the item with an id of 123456 even when applied multiple times.

7.3.2. API Usage

The web application makes use of the API to power all its core functionalities such as insertion, reading, deletion and editing. The entire data flow and security of the use-cases is controlled via the API. The web application relies on the API to provide with real-time information of the store activity concurrently, to visualize it as a heat map on the front end. The mobile application also makes use of the API to power several of its functionalities such as retrieval of all the in-app information, processing orders. The core user tracking and beacon's proximal interaction activity relies on the API to provide with real-time information from the DB to propagate and push to the web application.

A route method is derived from one of the HTTP methods, and is attached to an instance of the DB class. The following code is an example of how a route that is defined for the GET and the POST methods to the root of the app.

```
var router = express.Router();
// routes for APIs
router.route('/getData')
  .post(function(req, res) {
    // compose query to DB & process data
    res.json(reply); // reply the request
  })

// register routes, prefixed with /api
app.use('/api', router);
// start MongoDB client & the API server
DB.connect(url, option, function(err, db) {
  app.listen(def.server_port);
});
```

Route parameters are named URL segments that are used to capture the values specified at their position in the URL. The captured values are populated in the reply. JSON object, with the name of the route parameter specified in the path as their respective keys. The API contains methods are used to following routing methods over HTTP: get, post, put, head, delete, options, trace, copy, lock, mkcol, move, purge, propfind, proppatch, unlock, report, mkactivity, checkout, merge, m-search, notify, subscribe, unsubscribe, patch, search, and connect.

7.4. Implementation Process for Mobile Application

7.4.1. Triggering events when in a beacons proximal range.

```
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?
    //      let uuid = UUID(uuidString: "F7826DA6-4FA2-4E98-8024-BC5B71E0893E")!

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
        [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillResignActive(_ application: UIApplication) {
        // Sent when the application is about to move from active to inactive state. This can occur for certain
        types of temporary interruptions (such as an incoming phone call or SMS message) or when the user
        quits the application and it begins the transition to the background state.
        // Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks.
        Games should use this method to pause the game.
    }

    func applicationDidEnterBackground(_ application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers, and store enough
        application state information to restore your application to its current state in case it is
        terminated later.
        // If your application supports background execution, this method is called instead of
        applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(_ application: UIApplication) {
        // Called as part of the transition from the background to the active state; here you can undo many of
        the changes made on entering the background.
    }

    func applicationDidBecomeActive(_ application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application was inactive. If the
        application was previously in the background, optionally refresh the user interface.
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also
        applicationDidEnterBackground:.
    }

    func locationManager(_ manager: CLLocationManager, didChangeAuthorization status: CLAuthorizationStatus) {
        if status == .authorizedWhenInUse {
            if CLLocationManager.isMonitoringAvailable(for: CLBeaconRegion.self) {
                if CLLocationManager.isRangingAvailable() {
                    print("Starting Scan")
                    startScanning()
                }
            }
        }
    }
}
```

The following code is the app user interface delegate, that is used as the view controller of the application UI activity. This includes the functions; will resign active, did enter background, will enter for ground, did beacon active, did beacon became active and will terminate. What these functions perform are commented in the functions respectively.

The function location manager with the instance of didchangeAuthorisation , ensures that the location services and Bluetooth is enabled for the app to use, if the didchangeAuthorisationstatus is true , an instance of isRangingAvailable() method is triggered,

this starts the startScanning() function, else it will display warnings to turn location services and bluetooth.

```
func startScanning() {  
    let uuid = UUID(uuidString: DB.GET(beacon:UUID))!  
    let beaconRegion = CLBeaconRegion(proximityUUID: uuid, major: 0, minor: 2, identifier: "iBeacon")  
  
    beaconRegion.notifyOnEntry = true  
    beaconRegion.notifyOnExit = true  
    locationManager.startMonitoring(for: beaconRegion)  
    locationManager.startRangingBeacons(in: beaconRegion)  
}
```

The function StartScanning() starts to look for any beacons that is present in the region, the UUI for these beacons to scan for are pulled by the DB.GET object, if the beacon are found are found or when they are excited the functions notify the beaconRegion object.

Once a beacon has been found, an instance of startMonitoring and startRangingBeacon will be triggered, for the specific beacon. These two functions are used monitor and calculate the beacon region.

```
func locationManager(_ manager: CLLocationManager, didRangeBeacons beacons: [CLBeacon], in region:  
    CLBeaconRegion) {  
    if beacons.count > 0 {  
        print(beacons)  
        updateDistance(beacons[0].proximity)  
    } else {  
        print("No beacons in region")  
        updateDistance(.unknown)  
    }  
}
```

The instance didRangeBeacon is actively monitoring for the beacons that was found in the region, where the user is actively present in. if a change in the amount of beacon is found, the method updateDistance is called to update the proximity value of that specific beacon else the status of the proximity beacon is set to unknown.

```

func updateDistance(_ distance: CLProximity) {
    //print("Updating Distance")
    UIView.animate(withDuration: 0.5) {
        switch distance {
        case .unknown:

            if self.Unknown_current == self.Acualcurrent {

                let homepage_url = URL(string: DB.GET(URL))
                let homepage_request = URLRequest(url: homepage_url!)
                self.WebView.loadRequest(homepage_request)
                self.Unknown_current = "Active_Logged"

            }else{
                self.Far_current = "Active"
                self.Near_current = "Active"
                self.Immediate_current = "Active"
            }

        case .far:
            if self.Far_current == self.Acualcurrent {

                let homepage_url = URL(string: DB.GET(URL))
                let homepage_request = URLRequest(url: homepage_url!)
                self.WebView.loadRequest(homepage_request)
                self.Far_current = "Active_Logged"

            }else{
                self.Unknown_current = "Active"
                self.Near_current = "Active"
                self.Immediate_current = "Active"
            }

        case .near:
            if self.Near_current == self.Acualcurrent {

                let homepage_url = URL(string: DB.GET(URL))
                let homepage_request = URLRequest(url: homepage_url!)
                self.WebView.loadRequest(homepage_request)
                self.Near_current = "Active_Logged"

            }else{
                self.Unknown_current = "Active"
                self.Far_current = "Active"
                self.Immediate_current = "Active"
            }

        }
    }
}

```

The function `updateDistance()` is used to change the status of the beacon, and display what data is relevant to that specific beacons proximal range. There are four proximity zone a beacon can have:

- *immediate* (strong signal; usually up to a few centimetres)
- *near* (medium signal; usually up to a few meters)
- *far* (weak signal; more than a few meters)
- *unknown* ("hard to say", usually when the signal is very, very weak)

This method is running in the background, and concurrently and called every 5 seconds, once interaction controller classify the beacon's region, an API called is made to retrieve the data that is relevant to that beacons UUID and proximity zone, from the pulled data this displayed onto the Controller

7.4.2. Updating real-time location to the data base via API

```
func locationManager(_ manager: CLLocationManager, didEnterRegion region: CLRegion) {
    if let beaconRegion = region as? CLBeaconRegion {
        print("DID ENTER REGION: uuid: \(beaconRegion.proximityUUID.uuidString)")
    }
}

func locationManager(_ manager: CLLocationManager, didExitRegion region: CLRegion) {
    if let beaconRegion = region as? CLBeaconRegion {
        print("DID EXIT REGION: uuid: \(beaconRegion.proximityUUID.uuidString)")
    }
}

func alert(message: String, title: String = "Blue Tooth Status")
{
    let alertController = UIAlertController(title: title, message: message, preferredStyle: .alert)
    let OKAction = UIAlertAction(title: "OK", style: .default, handler: nil)
    alertController.addAction(OKAction)
    self.present(alertController, animated: true, completion: nil)
}
```

Each time and event interaction occurs, such as moving from one beacon to another or moving from a beacon's proximity region to another, the Location Manager updates the whereabouts of the user via the alertController, this pushes the data to the database via the Rest View controller.

7.4.3. Retravel of data from API

```
@interface RestViewController ()

@end

@implementation RestViewController

- (IBAction)fetchDATA;
{
    NSURL *url = [NSURL URLWithString:@"https://draj.me/JustDIY"];
    NSURLRequest *request = [NSURLRequest requestWithURL:url];
    [NSURLConnection sendAsynchronousRequest:request
    queue:[NSOperationQueue mainQueue]
    completionHandler:^(NSURLResponse *response,
    NSData *data, NSError *connectionError)
    {
        if (data.length > 0 && connectionError == nil)
        {
            NSDictionary *greeting = [NSJSONSerialization JSONObjectWithData:data
            options:0
            error:NULL];
            self.DataId.text = [[greeting objectForKey:@"id"] stringValue];
            self.Content.text = [greeting objectForKey:@"JASON.OBJ"];
        }
    }];
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self fetchGreeting];
}

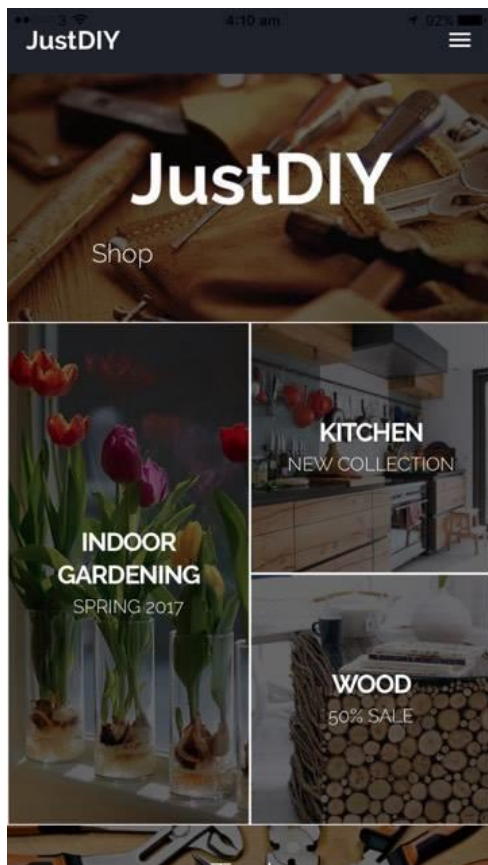
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

}
```

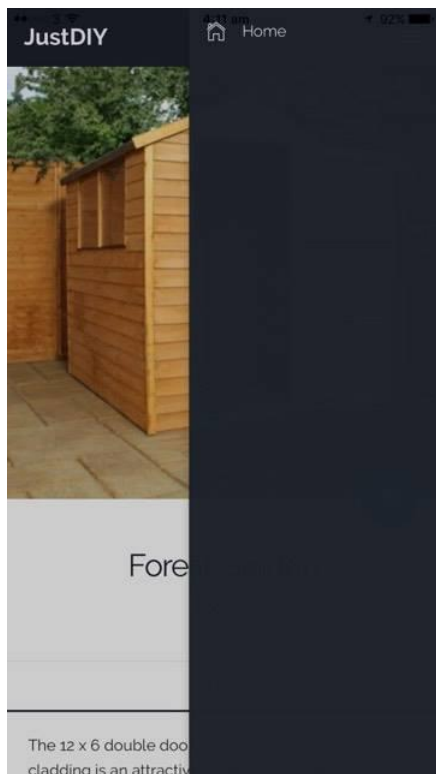
the following interface is used to retrieve the data that is stored on the API end point, the data returned from the endpoint is a JSON object along with the fragment id, this is then parsed to display the content on the UI Controller.

7.5. GUI implementation of the mobile app

As one of the aspects of the project to function as proposed is to create an interface for the mobile application. But as I before mentioned before the aim is to create the end-product will be a fully functional prototype so there will be more focus on meeting the functional requirements rather than having a visually appealing application. The but I still took the time to create a visually appealing and indurative application front-end that meets my user interface specification. Below are screenshots of the working application.



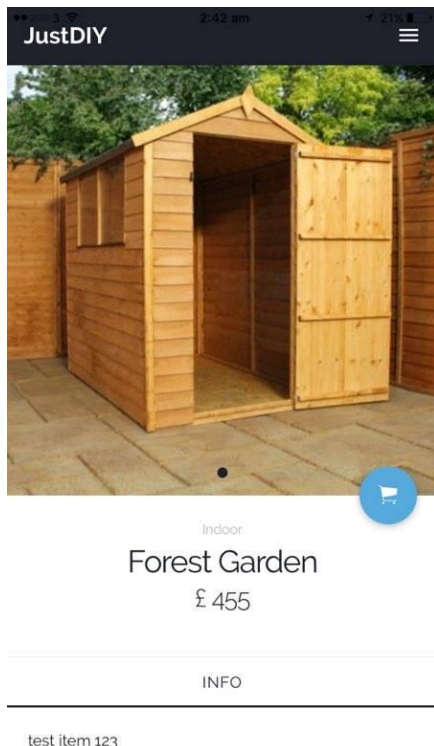
7.5.1. Homepage.



7.5.2. Navigation drawer.



7.5.3. Product category page



7.5.4. Detailed product view page

The screenshot displays the JustDIY mobile app's checkout page. The status bar at the top shows the time as 4:19 am and the battery level at 100%. The app header includes the 'JustDIY' logo and a hamburger menu icon. A 'Show order summary' link with a dropdown arrow is located at the top left, and the total price '£ 200' is shown at the top right. The page is divided into sections for 'Customer Information' and 'Shipping address'. The 'Customer Information' section contains an email input field with an envelope icon, a 'Login' link for existing users, and a 'Sign up' link for new users. The 'Shipping address' section includes input fields for 'First Name', 'Last Name', 'Company (optional)', 'Address', and 'Apt. suite, etc (optional)'. A 'Sign up' link is also present at the bottom of the page.

7.5.5. Confirmation page.

7.6. Development Process for Web Application

7.6.1. Adding a product to inventory

The screenshot displays the 'JustDIY' web application interface. At the top, the header includes the 'JustDIY' logo and the text 'Add a new listing'. A sidebar on the left contains navigation links: 'Overview', 'Inventory' (highlighted in red), 'Active HeatMap', 'Beacons', and 'Log Out'. The main content area features a form titled 'Add a new item' with the instruction 'Please fill the fields below in order to list your item.' The form includes the following fields: 'Product Name' (text input), 'Product URL' (text input), 'Price' (text input), 'Category' (dropdown menu with 'Sheds' selected), and 'Product description' (text area). A red button labeled '< BACK TO INVENTORY' is located at the top right of the form area. A green button labeled 'ADD ITEM' is positioned at the bottom of the form.

This is the frontend of the 'add item to the inventory' form that the shop owner will use in order to add an item to the inventory, once the user has filled in the required information, and pressed the add item button, a POST request is made to the API to insert the data into the database.

```

<form class="config-form" method="POST" action="add_item.php">
  <div class="config-block">
    <div class="row">
      <div class="col-lg-12">
        <label>Product Name</label>
        <div class="input-block">
          <input placeholder="My product name...." name="ProductName" id="Product_name" type="text">
        </div>
      </div>
      <div class="col-lg-12">
        <label>Product URL</label>
        <div class="input-block">
          <input placeholder="Product URL" name="URL" id="URL" type="text">
        </div>
      </div>
      <div class="col-lg-6">
        <label>Price</label>
        <div class="input-block">
          <input placeholder="Price" name="Price" id="Price" type="text">
        </div>
      </div>
      <div class="col-lg-6">
        <label>Category</label>
        <div class="input-block">
          <select name="Category" title="select page" class="custom-select">
            <option value="Sheds"> Sheds</option>
            <option value="Indoor"> Indoor</option>
          </select>
        </div>
      </div>
      <div class="col-lg-12">
        <label>Product description</label>
        <div class="input-block text-block">
          <label for="txt-about">Description</label>
          <textarea name="Description" id="txt-about"></textarea>
        </div>
      </div>
    </div></div></div>
    <div class="panel-footer">
      <div class="config-block">
        <button class="btn btn-success mt-0">Add Item</button>
      </div>
    </div>
  </div>
</div>
</div>
</form>

```

The HTML code that is used to render the front end form.

```

#Function to add item to inventory
if(isset($_POST['ProductName'])) {
$query = "INSERT INTO `Products` (`ProductName`,`Category`,`Description`,`URL`,`Price`) VALUES ('$
  ProductName','$Category','$Description','$URL','$Price')";
$result = mysqli_query($con, $query);
header('Location: inventory.php');
}

```

Once the isset for the POST request is triggered, all the parameters that were passed from the form is inserted into the database.

7.6.2. Displaying all products from the inventory

```
$sql = mysqli_query($con, "SELECT * FROM Products");  
if(mysqli_num_rows($sql)>0) {  
while($row = mysqli_fetch_array($sql)) {
```

The following code selects all the product records and its attributes that is found in the database, then the selected records are iterated and displayed on the frontend.

```
<td><div class="title-col">  
  <label for="item">  
    <?php echo $row['ProductName']; ?>  
  </label>  
  
  </div></td>  
<td><div><?php echo $row['Category']; ?></div></td>  
  
<td><div><?php echo $row['Price']; ?></div></td>  
  
<td><div class="last-col prod-actions">  
  <a href="edit_item.php?item=<?php echo $row['ID']; ?>">edit</a> <a  
    href="inventory.php?delete=<?php echo $row['ID']; ?>">delete</a>  
  </div></td>  
</tr>  
  
<?php } ?>  
<?php }  
  
else{ echo "You have no products in the inventory"; } ?>  
</table>
```

The HTML code that is used to render the front-end form. As you can see I have echo statements within the table, this is used to display the attributes of the records that is selected from the database.

JustDIY		Inventory			
		+ ADD PRODUCT			
	Product Name	Category	Price	Actions	
<input type="checkbox"/>	Forest Garden	Sheds	200	EDIT	DELETE
<input type="checkbox"/>	Indoor Table3	Sheds	233	EDIT	DELETE
<input type="checkbox"/>	Forest Garden	Indoor	455	EDIT	DELETE

Visual output of the form

7.6.3. Editing a product's details in the inventory

```
$ProductName = $_POST['ProductName'];
$Category = $_POST['Category'];
$Description = $_POST['Description'];
$URL = $_POST['URL'];
$Price = $_POST['Price'];
$ITEM_ID = $_POST['item'];

#Function USED

if(isset($_POST['ProductName'])) {
$query = "UPDATE Products SET ProductName='$ProductName', Category='$Category', Description='
$Description', URL='$URL' , Price='$Price' WHERE ID=$ITEM_ID";
if (mysqli_query($con, $query)) {
echo "Record updated successfully";
} else {
echo "Error updating record: " . mysqli_error($con);
}

header("Location: inventory.php");
}
```

Once the isset for the POST request is triggered, all the parameters that were passed from the editing form is updated in the database of the specific item with that ID. Once updated, a success message is displayed, then the user is navigated to the inventory page.

7.6.4. Deleting a product in the inventory

```
#To Delete a product from DB
if(isset($_GET['delete'])) {
$Del_ID = $_GET['delete'];
$query = "DELETE FROM Products WHERE ID='$Del_ID'";
mysqli_query($con, $query);
}
```

When the GET request for delete is triggered from the inventory page, The ID of that product is passed to the deletion query. Once the query is run, a success message is displayed.

7.6.5. Deleting a beacon from the system

```
#To Delete a beacon from DB
if(isset($_GET['delete'])) {
$Del_ID = $_GET['delete'];
$query = "DELETE FROM beacon WHERE ID='$Del_ID'";
mysqli_query($con, $query);
}
```

When the GET request for delete is triggered from the inventory page, The ID of that product is passed to the deletion query. Once the query is run, a success message is displayed.

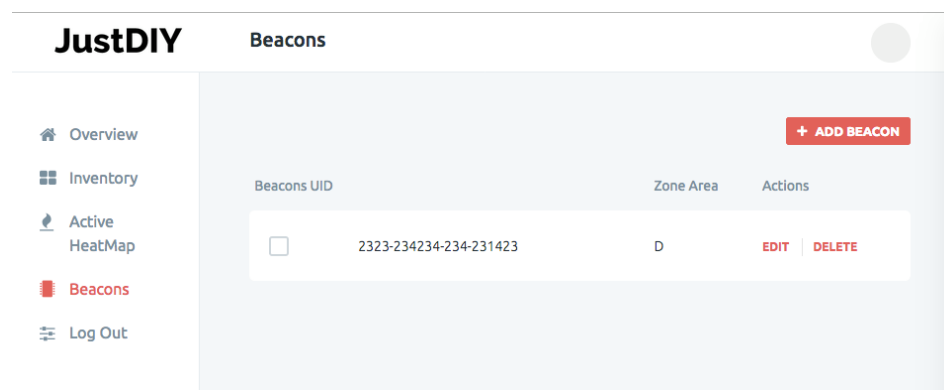
7.6.6. Displaying all beacon in system

```
$sql = mysqli_query($con, "SELECT * FROM beacon");  
if(mysqli_num_rows($sql)>0) {  
while($row = mysqli_fetch_array($sql)) {
```

The following code selects all the beacon records and its attributes that are found in the database, then the selected records are iterated and displayed on the frontend.

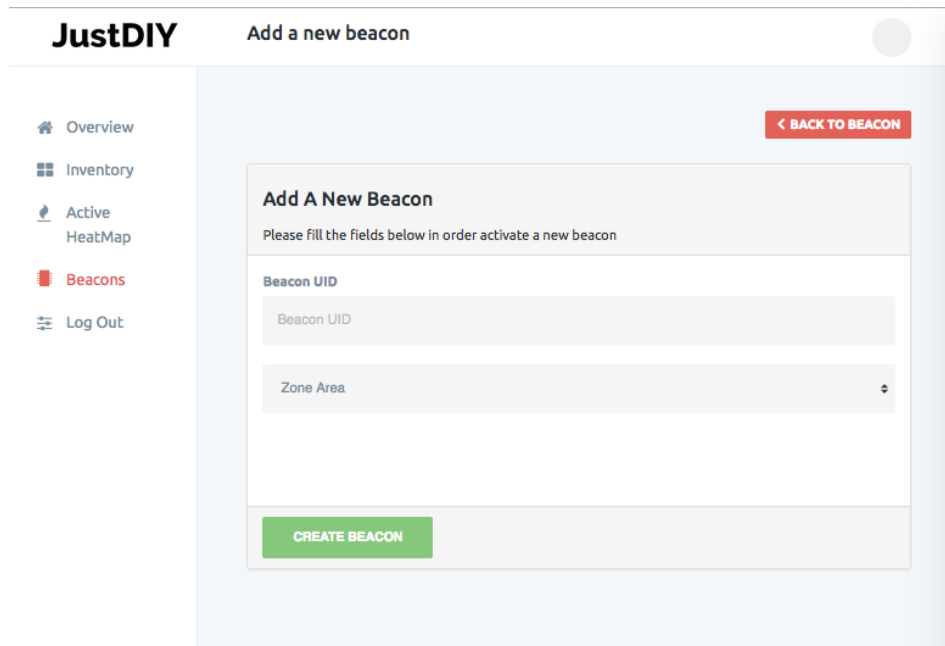
```
<tr>  
  <td><div class="title-col">  
    <label for="p011">  
      <?php echo $row['UID']; ?>  
    </label>  
  </div></td>  
  <td><div> <?php echo $row['Zone']; ?></div></td>  
  
  <td><div class="last-col prod-actions">  
    <a href="edit_beacon.php?item=<?php echo $row['ID']; ?>">  
      edit</a> <a href="beacon.php?delete=<?php echo $row['ID'];  
      ?>">delete</a>  
    </div></td>  
</tr>  
<?php } ?>  
<?php } else { echo "You have no data to display"; } ?>  
</table>
```

The html code that is used to render the front-end form. As you can see I have echo statements within the table, this is used to display the attributes of the records that is selected from the database.



Visual output of the form

7.6.7. Adding a beacon to the system



The screenshot shows the 'JustDIY' web application interface. The header includes the 'JustDIY' logo and the title 'Add a new beacon'. A sidebar on the left contains navigation links: Overview, Inventory, Active HeatMap, Beacons (highlighted in red), and Log Out. The main content area features a red button labeled '< BACK TO BEACON' at the top right. Below it is a form titled 'Add A New Beacon' with the instruction 'Please fill the fields below in order activate a new beacon'. The form contains two input fields: 'Beacon UID' and 'Zone Area'. At the bottom of the form is a green button labeled 'CREATE BEACON'.

This is the frontend of the 'add beacon to the system' form that the shop owner will use in order to add a beacon to the system, once the user has filled in the required information, and pressed the add item button, a POST request is made to the API to insert the data to the database.

```
#Function USED

if(isset($_POST['UID'])) {
    $query = "INSERT INTO `beacon` (`UID`,`Zone`) VALUES ('$UID','$ZONE')";
    $result = mysqli_query($con, $query);
    header('Location: beacon.php');
```

Once the isset for the POST request is triggered, all the parameters that was passed from the form is inserted into the database.

7.6.8. Editing a beacon's details in the system

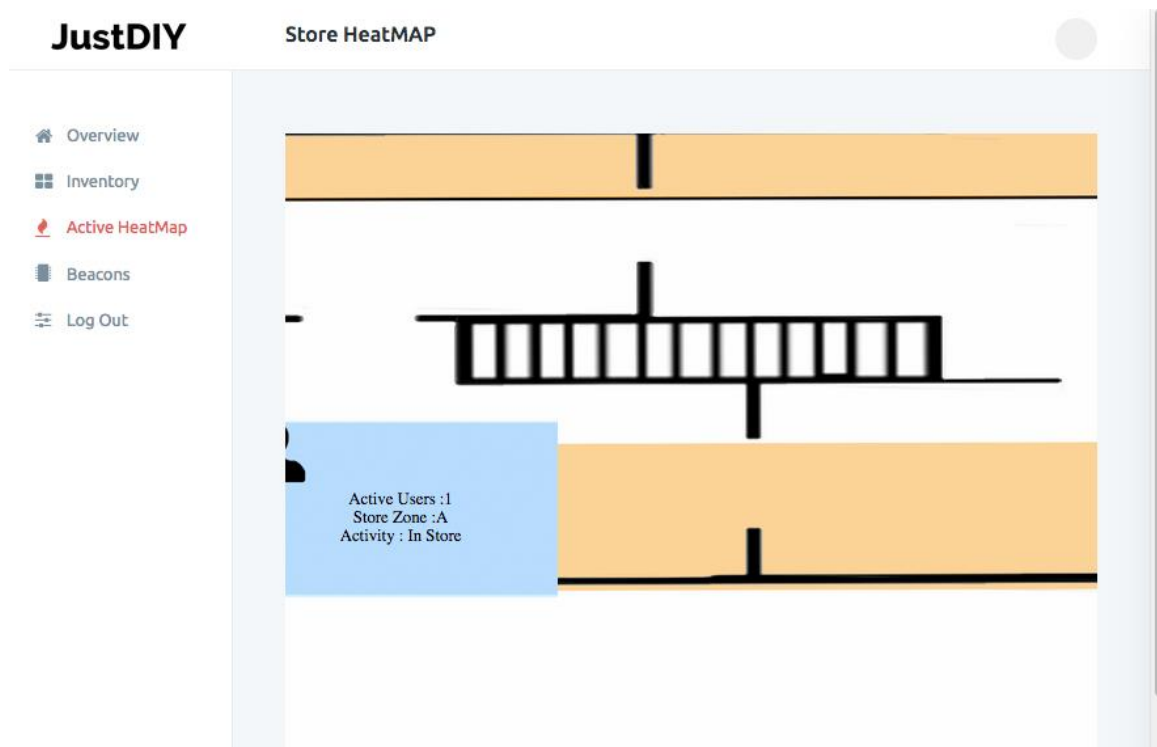
```
#POST Pram
$UID = $_POST['UID'];
$ZONE = $_POST['ZONE'];
#Function USED

if(isset($_POST['UID'])) {
$query = "UPDATE beacon SET UID='$UID', Zone='$ZONE' WHERE ID=$ITEM_ID";

if (mysqli_query($con, $query)) {
echo "Record updated successfully";
} else {
echo "Error updating record: " . mysqli_error($con);
}
header("Location: beacon.php");
}
```

Once the isset for the POST request is triggered, all the parameters that were passed from the editing form is updated in the database of the specific item with that ID. Once updated, a success message is displayed, then the user is navigated to the beacons page.

7.7. Displaying the active heat map of store floor activity



This is the front end of that is used to display the heat map. The heat map is used to display the active users in the store in a visual representation. This page makes a concurrent request every 5 seconds so that it will display the real-time data on the page. The heatmap shows which zone and what activity is a specific user is doing real time. The congestion of a zone is displayed by shades of colors ranging from red to blue. The entire area of the heat map is an HTML canvas element so when a change is made, such as a new user in a different zone or more users in at the same zone, the page auto updates the data with an AJAX call, then overwrite the content with the latest data.

7.7.1. The concurrent AJAX call to update the canvas

```
function HeatMapCanves() {  
    $.HeatMap();  
}  
  
window.onload = HeatMapCanves;  
window.setInterval(function(){  
    $.HeatMap();  
}, 5000);
```

The index page contains the following code, the function creates an instance of HeatMapCanves() then calls it concurrently every 5 seconds, load the new data onto the canvas.

```
$(document).ready(function() {  
    $.HeatMapCanves = function(settings) {  
        $('<div id="heatmap-overlay"></div>').appendTo('body');  
        var value = {  
            'page':escape(document.location.pathname)  
        };  
        $.ajax({  
            url: "heatmap.php",  
            type: 'GET',  
            data: value,  
            dataType: "text",  
            cache: false,  
            success: function(response) {  
                $(response).appendTo('body');  
            }  
        });  
    };  
});
```

Heatmap.js has the function HeatMapCanves() which when called upon makes a GET request to the heatmap.php page, this page is then used to load and layer the new data that has been pulled from the database.

```

//when the display heatmap Js file is called VIA GET
if(isset($_GET['page'])) {
    $query = "SELECT * FROM heatmap";
    $results = mysql_query($query,$db);
    // Get the data That is need to appen on to the canves.
    $html = '<div id="heatmap-container-blue">';
    $Count_A = 0;
    $Count_B = 0;
    $Count_C = 0;
    $Count_D = 0;
    $Count_E = 0;
    $Count_F = 0;
    while ($row = mysql_fetch_array($results)) {
        $maincount = 0;

        if ($row['zone'] == "A") {
            $Count_A += 1;
            $maincount = $Count_A;
            $Area = "Store Zone :A <br>Activity : In Store";
        }
        elseif ($row['zone'] == "B") {
            $Count_B += 1;
            $maincount = $Count_B;
            $Area = "Store Zone :B <br>Activity : In Product Area";
        }
        elseif ($row['zone'] == "C") {
            $Count_C += 1;
            $maincount = $Count_C;
            $Area = "Store Zone :C <br>Activity : Viewing Product";
        }
    }
}

```

The following code is what pulls all the real-time data stored in the database, then it sorts the data according to the beacon area zone, once all the users are counted it is returned back to the data to be omitted on to the page.

```

#The Colour Switching FUNCTION
if ($maincount <= 2) {
    $colour = "blue";

    $type = 'background:transparent url(images/click.png) no-repeat center center; ';
}
elseif ($maincount <= 4) {
    $colour = "light Orange";

    $type = 'background:transparent url(images/orange.png) no-repeat center center; '
    ;
}
else {
    $colour = "Red";
    $type = 'background:transparent url(images/red.png) no-repeat center center; ';

    # $html .= sprintf('<div id="heatmap-container-red">');
}

$html .= sprintf('<div style="left:%spx;top:%spx;'. $type. '">

    <center><br><br><br><br><br><br> Active Users :'. $maincount. '<center> '. $
    Area. '</center></div>');

}
$html .= '</div>';

echo $html;

```

Once user has been counted, each zone is classified per the congestion of active users, present at a given time of a zone, this is displayed by shades of colors ranging from blue to red, This is then layered on top in the zone section of the heat map to be omitted on to the page.

7.8 Implementation challenges

7.8.1 Problems Encountered

Challenges were also encountered when trying to get the mobile application to retrieve data from the API, this meant that certain data streams were not parsed correctly on the front end and thus it would break the front-end elements when an empty string was returned. To fix the problem, if a GET request was failed, it would use dummy filler data to handle the events gracefully.

7.8.2 Final Interface Design Changes

Some parts of the interface design were changed to some extent during implementation. Nevertheless, the overall interface design was showed to be very effective. The changes that were made were very minor modifacations to the user interface specification. The web application was planned to be as simple and effective as possible and research was carried out to build a solid user interface. Due to time constraints, some aspects of the design were not perfect as planned in the design, such as the heat map. The user activity data was displays well on the page however it was hard to fit all the collected user data on the page and this is something that could be changed in the future.

8. Testing

This chapter will discuss the overall approach and results of testing carried out on the tracking algorithm, mobile application, and web application. Due to the functional development style of the project, a detailed test planning and specification document was not done at the initial design stage.

8.1. Overall Approach to Testing

As mentioned in the design section, PHPUnit test were used to test the PHP code and QUnit for testing the JavaScript functionalities of the web application. Different kinds of tests were performed on the mobile and web application and it involved usability, use case, interface, functionality, performance and compatibility testing. Testing was carried out to detect if the tracking algorithm and the applications were functionally correctly. This involved carrying out a set of tasks and comparing the test results with the expected outcome of the functionality. The tests were carried out several times with different data input to check the accuracy of the algorithm and system. Some testing was done at each phase of development to speed up development and to improve the quality which reduced the risks towards the end phase of development. To deliver an accurate and effective tracking algorithm, mobile and web application, it was important to identify any bugs within the system during the testing.

The initial aim was to write and run QUnit and PHPUnit tests before coding for the client side and server side applications but due to the ambiguities on how some of the tests should be written, tests were left until the coding stage was completed, for some functionalities of the system. Some functionalities of the application were quite complex to test, as it requires setting up a physical environment to test how efficient and fast the tracking algorithm performed in the physical environment parameters. Tests were also written for a specific proximal range of an event that is prompted with data having to be taken from the API to get the information from the server was complex and hence slowed the development process.

8.2. Use Case Testing

Use case testing phase was carried out by various student volunteers by testing the mobile application and system infrastructure. This was done to check if the system met the use case requirements and if the mobile application reacted as intended when entering different beacon zones and provided the activity information to the web application. Users were given a feedback form which had questions asking about their opinion on the mobile application and how it functioned. The feedback form is added in Appendix B. Use case testing helped in analyzing the usability of the mobile application and provided me with sufficient information and data to determine the success of mobile application and tracking algorithm from a user's point of view. The feedback provided by the users was useful in identifying the applications anomalies and unexpected interaction behavior. The results of the acceptance tests had some positive feedback and had some improvements mentioned. The bugs reported by the users

were taken into consideration and was fixed but the addition of extra functionalities was not a feasible option due to the time constraints and this is something that can be improved in the near future.

8.3. Functional and Interface Testing

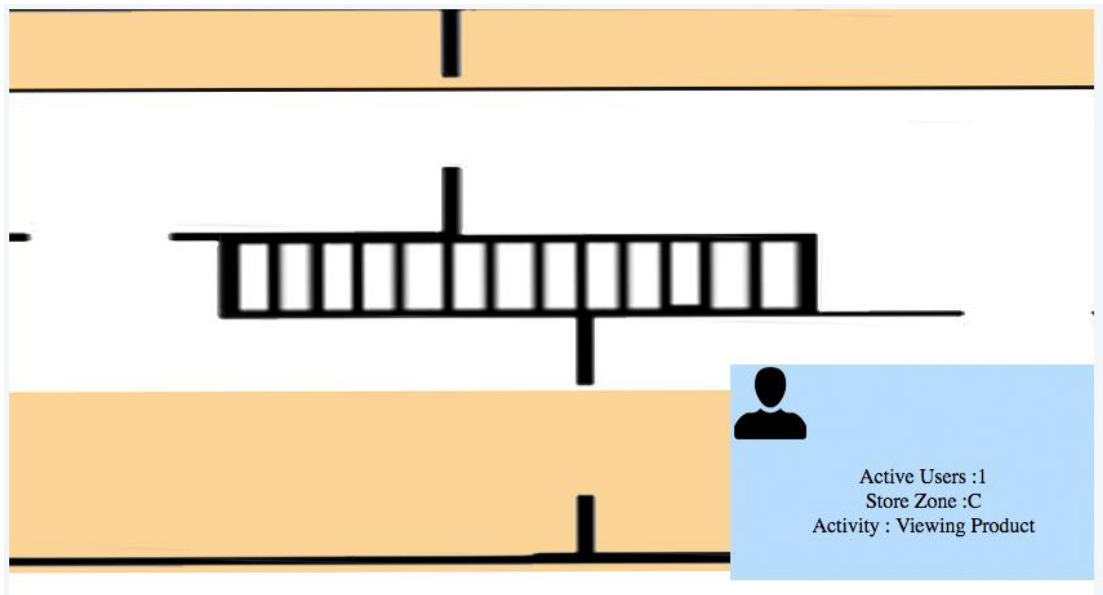
Interface testing was carried out to check if the interactions and data transfer between the mobile and web application and the API server was working as expected. It was also done to make sure that the queries sent to the database via the API gave expected results and to ensure that the errors were managed gracefully. Functional testing was carried out to check if the applications were as performing all the criteria that were set in the initial specification requirements of the intended system. Functional tests involved checking the usability of the system since the navigational functions of the interface were tested on the mobile and web application. Several tests were carried out to ensure that the features of the applications were working as expected. The testing table for the interface and functionality tests are included in Appendix B.

8.4. Automated Testing

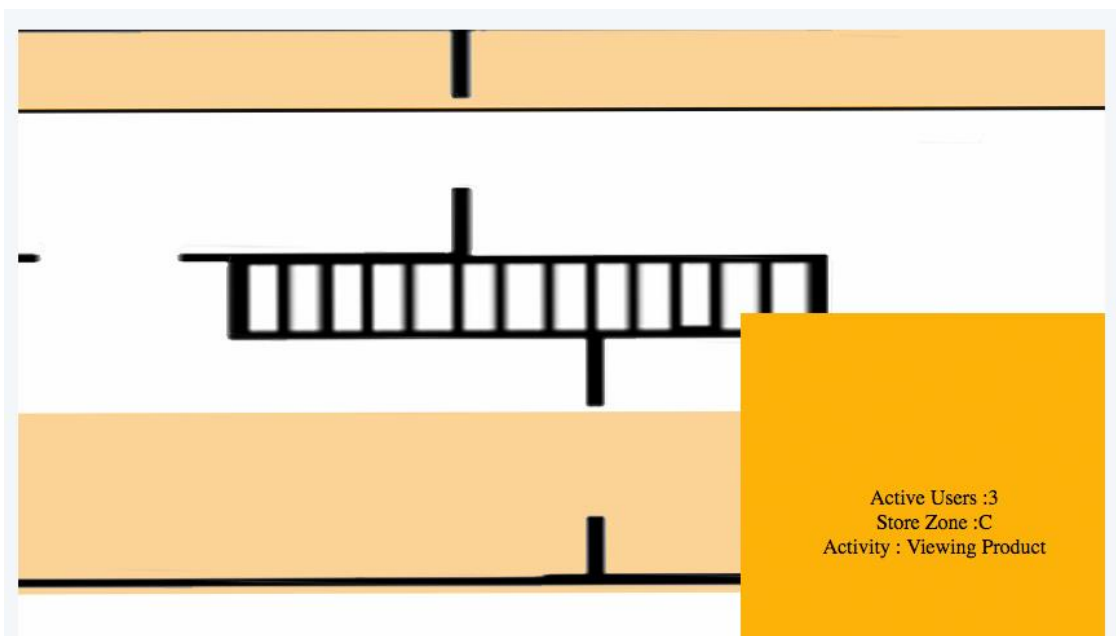
PHPUnit was used to test the quality of the code. To run automated tests on the browser, cucumber had to be installed with PHPUnit. Installing cucumber with PHPUnit was a big challenge for me. Many attempts were carried out to fix the problem by reinstalling different versions of PEAR (PHP Extension and Application Repository) but were not successful in installing. As time was a huge constraint, I decided to carry out acceptance tests, in the end, to check if the system functioned as proposed in the initial specification of the intended system. Even though there are many benefits in carrying out automation tests, the main purpose of the system is the functional user tracking in the system, this was dedicatedly tested. In the future, automated testing is an area that needs to be considered, as it will enable us to test the system robustly and raise the quality of the system.

8.5. Algorithm Testing

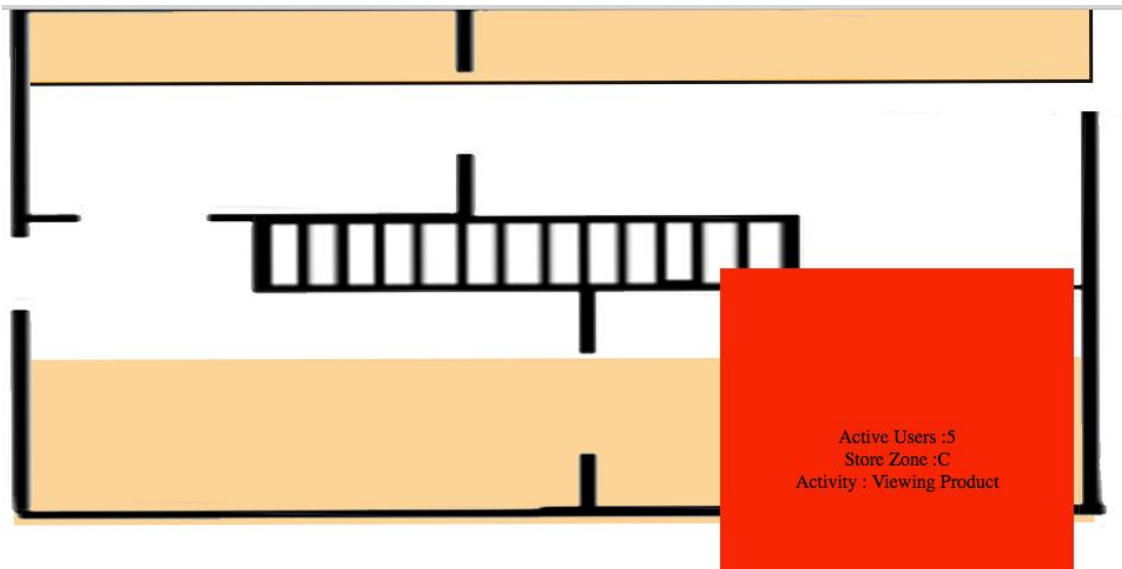
In order to test if tracking algorithm was working as intended to, I had to physically create a store area and deployed beacons in a dedicated zone of the room. This allowed me to simulate a real-life scenario on how the tracking algorithm performed as users interact with different areas of the store. For this experiment, I created 6 real-life scenarios that can happen. The output from the real- time heat as follows. The test results obtained on how the phone reacted on different proximal rage can be seen in appendix A.



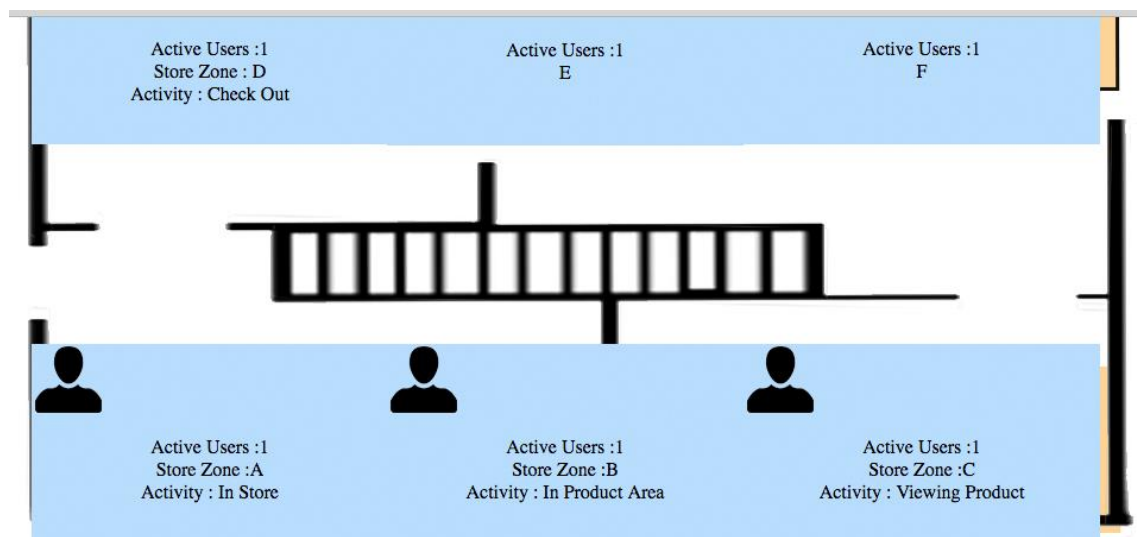
One active user at zone C of the store.



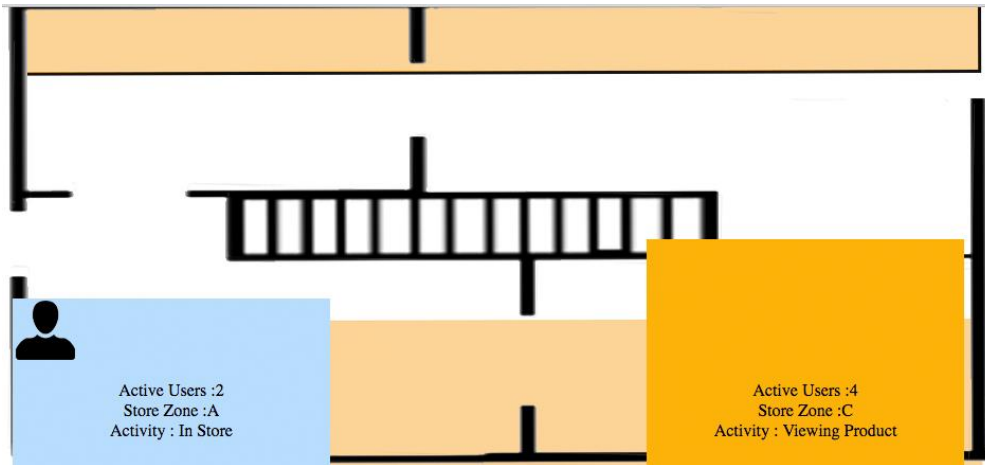
Three active users all at zone C of the store.



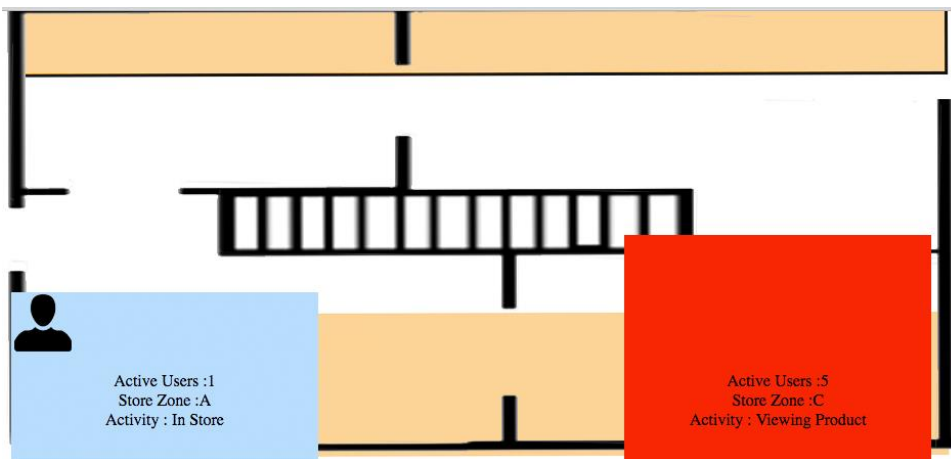
Five active users at zone C of the store.



Six active users, all at the different zone of the store.



Four active users at zone C and two active users at zone A of the store.



Five active users all at zone C and two active users at zone A of the store.

From the experiments that I conducted on the tracking algorithm and the heat map, I can conclude that the system is working as intended to. The congestion of active users, present at a given time of a zone is displayed by shades of colors ranging from blue to red, from these tests carried out I can conclude that the functionality is also working as intended to.

8.6. Test Result

The test results from the use case testing and functional and interface testing can be seen in the appendix A.

9. Evaluation

In order to measure to what extent, the system and infrastructure have achieved its initial objectives, and to know the limitations, the system capabilities and overall project characteristics are evaluated. Looking back at my initial goals, the system has achieved most of the required functionalities and has provided a usable system experience to the end user. Overall the system and infrastructure were successful in meeting the required goals of the project.

The system and infrastructure also provide a strong platform for extending it in the future for a commercial product. It provides a strong base for further advancement in the development of large ad-hoc beacon network & providing functionalities that allow developers to interact and collect user data in real-time.

Due to the limited amount of time, physical space, and hardware accessibility constraints, integrating a large beacon network was not a feasible option. Even though the system has so many positive aspects, there are also some weaknesses and limitations which can be improved to make the project have a commercial viability and highly scalable in the future. This chapter examines and evaluates the successes and drawbacks of the project in depth.

9.1. Functionality Fulfilment

Overall an efficiently working system which performs the functional requirements of the project was fulfilled and was working well as expected. The responses that I got back from the testing were positive which meant that the end users were happy with the system.

It would have been good to have a larger group of users to test the mobile application to get more feedback, on how the system performance on different loads and multiple users at a given time. Additionally some advanced functionalities were suggested by volunteers from acceptance test, but this was not possible to add due to lack of time and this is something that has to be taken into consideration in the future development stage.

9.2. Project Methodology and Management

In my opinion, adhering to the Feature Driven Development methodology was sufficient to some extent for the project. Strictly following the intermediate milestones which were agreed at the project planning stage ensured a steady delivery of the functionalities being completed on time. A great emphasis was placed on designing and modeling in FDD and this was an advantage since the requirements and complexity of the functionalities were then analyzed before the implementation phase. Carrying out preliminary research work during the design stage helped in understanding the complexities of the project before the implementation

phase. This made it easier to construct and deploy the solution during the implementation stage.

Even though most of the complexities of the project were examined at the design phase, there were still some problems during implementation. For example, push notification to the phone, while the phone is locked was thought as not to be that difficult during the design stage. While implementing, it was found that interacting with a locked phone was complex and more research was needed and that it was time-consuming which slowed the overall development of other functionalities of the system. Eventually, it was not successfully implemented due to lack of time and the amount of the research needed but this is certainly an area of modification that can be done in the future.

Another major problem was found while writing test code before the implementation stage. The initial aim of FDD was to write unit tests code before the implementation since it forces you to plan and think about the design before you code and therefore it would make the design better for the project. However, due to the ambiguities on how some of the tests should be written, tests were left until the coding was completed for some functionalities of the system. Although the functionalities of the system were successfully tested after implementing by functional testing, unit testing is something the system lacks and certainly needs improvement in the future for better code quality.

Due to the iterative nature of Feature Driven Development and adequate design upfront, it was easier to adapt to the changes at the design stage which made the implementation process easy. Coding in FDD seemed like a mechanical process which was a drawback during implementation. Also, refactoring code is something FDD discourages. At first, I thought refactoring code was not necessary but as the project went along, it was found necessary to refactor the code since it was easier to support the development and changes in the future.

Having chosen to do OOP programming style in PHP was also beneficial for the project as it enabled me to build loosely coupled system components, thus ensuring in a later date when changing major module or adding a new module in a class should not affect the others. Following these standards ensured that the code that I produced was highly extensible and modular, this essential as it allows wider scope for future development.

9.3. Technology and Design Evaluation

The technology and design choice that I made for the system has shown to produce an effective and successful solution for the end product. This following section will analyses how successful the technology choices are for delivering the final solution.

9.4. Web technology

Most of the web application interaction functionality was created out using JavaScript and AJAX. Having decided to choose JavaScript, AJAX and HTML5 was certainly a good choice for the client side technology. The choices helped in making the website web application much more interactive and responsive and thus provided a robust and efficient solution for the front end.

The entire area of the heat map is a HTML canvas element so when a change is made, from an AJAX call, such as a new user in a different zone or more users in at the same zone, the page auto updates then, then overwrite the content with the latest data. Using AJAX allowed me to do this task. I also used Bootstrap for the project since it helped rapidly develop UI components, thus speeding up the development process.

9.5. Mobile Technology

The iPhone app implementation was easier than what I expected as apple provided an iPhone IDE and emulator tools, this enabled me to test the app on the go as I was developing each feature. Apple also provided with built-in hardware accessibility library, which enabled me to access the Bluetooth receiver to implement, the interaction controller, that responds to the beacon signals. The language that I selected to develop for iOS was Swift, as I have never used Swift before, I had to under how some extensive training to get working proficiency, this took some time, but after undergoing the training I was able to start developing the mobile application without any problems. At some case, I did not know how to implement certain elements, but apple provided with very good developer documentation and a developer community to resolve this problem.

9.6. Server-Side Technology

Choosing the server side language was thoroughly thought out at the design stage to analyses if the language provided the necessary tools that were essential for developing the project. Choosing PHP was a great language choice for developing the backend system and API because of the various tools and libraries PHP offered and the great developer community support and well-documented code eased the development process. Selecting PHP was also advantageous as I am very familiar with the language prior, as I have used in other projects.

9.7. Data Management

Having a good database design was a vital part of the development process in order to deliver high performance and availability. Selecting MySQL was an advantage because it was easy to perform queries and it also provided a secure storage for the data. Overall, it worked well with the API and the performance of the data interaction well with external entities.

9.8. Software Design Evaluation

The design decisions and data structure choices made were successful to some extent for the project. Choosing object oriented programming for the project made the code maintenance easier since each function was designed to perform certain jobs. Also, the functions do not depend on any external state which makes adding extra functionalities easier in the future. Overall the system and hardware architecture was decoupled which is an advantage since it makes the code reusable and adding additional functions and modifications can be done easily in the future. Overall the project software design was successful for the project.

9.9. Hardware Evaluation

Primarily the Implementation phase of the project went fairly well even though certain parts of the project took a longer time to implement than expected. Some functionalities of the system were not successfully implemented as planned but overall, most of the functional requirements were met. Code quality was maintained throughout the implementation process. Refactoring is something that can be done to improve the code so changes are much easier to implement in the future. The development tools chosen for the implementation helped in managing the project well but managing time was an issue during the implementation stage. Even though features on the system were developed in the order planned out, implementing certain features on the system took longer than expected. Also, fixing some problems took a long time and it was hard to manage the time spent on problems which slowed the overall development to some extent. A project management tool which monitors time spent on each task could have helped to manage time better.

9.10. Implementation Evaluation

Initially, the Implementation stage of the project went fairly well even though certain parts of the project that took a longer time to implement than expected. Some functionalities of the system were not successfully implemented as planned but overall, most of the core functional requirements were met. Code quality was maintained throughout the implementation process. Refactoring is something that can be performed to improve the code quality even further, so it is easier to support the development and changes in the future.

The development tools I selected for the implementation stage leveraged and aided in managing the project well and lowered the risks, but managing time was an issue during the implementation stage. As I had a few setbacks to get the hardware to working as intended to, even though features on the system were developed in the order planned and then implementing, certain features on the system relies on the beacon to function is such way, to get it working took longer than expected. Also, fixing some mobile and web application problems took a long time and it was hard to manage the time spend on each problem which slowed the overall development to some extent. A project management tool which monitors time spent on each task could have helped to manage time better and evaluate risks.

9.11. Testing Evaluation

Carrying out functionality and acceptance testing has proved that the system is functionally working as intended to, but the lack of unit and automation testing is a drawback of the system. This is an area that can be further improved which can help in identifying the edge cases of the code in more detail which aid in increasing the quality and robustness of the system.

9.12. Future Development

There are many ways in which the project can be improved and further developed in the future and there are many aspects of this project changed to create a commercially viable product. Several more functionalities can be added on to the system to make it more desirable. Some of the features as follows.

- **Instore navigation help**

The ability to show the where about of the user inside the mobile application. So, the idea was the mobile users were able to see on the floor map and pinpoint where they are in the physical space. So, they can use this information to navigate to different areas of the store. This is an innovative idea which can be an advantage since it can help users discover areas while cutting down staff time that is used in interaction for help previously.

- **Time logging**

From store floor interaction, collect user data on how long a user stayed at a beacon's proximal range, from on-enter to on-exit of the proximal region. This data will allow us to perform data mining on at a later date to deduce what areas are most popular and create a user time log, which contains which path the user took and how long they stayed in a specific area for. This can be hugely beneficial analytical data that the shop owner can use, to perform strategic marketing campaigns and promotions specific to that user.

- **Push notification**

This feature would allow shop owners to push popup notifications from the web backend panel to the mobile phone, this can be used by shop owners to target buyers with promotional offers.

- **Build the app on android**

In the future, the mobile application should be also be built for android, so all users that are using the android phones can also make use of the system.

- **Detailed statistics**

Collect more analytical data, which can be used to at later date to deduce what areas are most popular, what are the peak times and create a user travel log, which contains which path the user took and how long they stayed in a specific area for. Having this vital information, it enables them to make, smarter and strategic business decisions.

- **Further user scalability testing**

This involves testing the system with many users at one given time, this will test the scalability of the system much more thoroughly and give feedback on how well it performs as the load increases.

9.13. Conclusion

The main aim of this project is to explore the use of beacon technology, in order to provide an immersing and targeted experience for shoppers, and allowing shop owners to collect meaningful real-time data from store floor activity, which can then aid them to make smarter and strategic business decisions. For the most part, this has been successfully fulfilled by the system.

In order to satisfy the goals of the project, I was able to develop an IOS mobile application that can be used by shop customers, and a back-end software and hardware infrastructure that allows beacons to transmit targeted advertisements to nearby users. I was also able to create the low voltage transmitting beacons from Raspberry Pi's and Bluetooth modules. A web application was built which will allow the shop owner to add new products to the inventory, create a new beacon fenced area and view real-time analytics of store floor activity.

Furthermore, with this application and its hardware, the biggest goal was to be able to have hi extensible and modularity. Which enabled the users to create and manage large ad-hoc beacon network & providing functionalities that allow developers to interact and collect user data real-time.

This project would also have high commercial value if further developed with more desired features, and further developed to integrate with other mobile operating systems since there are no current software available in the market that can do so.

The interface design of the application can be improved to make it more effective and user engaging. One of the improvements suggested from the acceptance test was the able to show the where about of the user inside the mobile application. So, the idea was the mobile users were able to see on the floor map and pinpoint where they are in the physical space. So, they

can use this information to navigate to different areas of the store. This is an innovative idea which can be an advantage since it can help users discover areas while cutting down staff time that is used in interaction for help previously. There are many more ideas that can be used to further develop and improve the system.

I found this project very challenging and enjoyable to work on since it was something I had never worked on and felt this is very interesting research sector. It took me a while to construct the beacon and get it to start emitting data, as I did not have any prior knowledge working with raspberry pi's this made the task very challenging. Therefore, the learning curve of the project was steep because I found the initial stage of development very daunting. Even though there were many problems at the start of the project, as the project went along I got better at understanding and eventually managed to complete the initial goals of the system.

One of my other biggest learning curve for this project was developed the iPhone app, as I have never developed an iPhone app before, I took on this challenge by taking programming courses in iOS development in Swift on Udacity, that was made by industry leaders from Google and Airbnb. After completion of the online course, it enabled me to become familiar with almost everything I needed to develop an app using Swift. Taking these courses further enabled me to build apps and empowered me to further design and develop my project app, with much better functionalities and user experience.

The project gave me a huge insight into the construction of Bluetooth beacons, tracking algorithms, statistical analysis, iOS & web deployment and much more technology's which would be a great advantage in the future and work industry. I have no doubt in concluding, working on this project has enhanced my technical skills immensely, as well as improved my existing project management and software development skills.

If I was to do the project again I would try to make the mobile application much more user friendly and immersive, as currently, I built the app to create fully functional prototype so the main focus was on meeting the functional requirements rather than having a visually appealing application, this resulted in having a slightly poor UI and UX for the user, so when developing it again there should be a strong emphasis on user-driven development.

10. Annotated Bibliography

- [1] Anon 2017. Available at: https://www.apple.com/business/docs/iOS_Security_Guide.pdf [Accessed: 5 May 2017].
- [2] Anon 2017. Available at: <https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf> [Accessed: 5 May 2017].
- [3] Anon 2017. Available at: <http://getbootstrap.com/2.3.2/index.html>. [Accessed: 5 May 2017].
- [4] Anon 2017. Available at: <http://spectrum.ieee.org/telecom/wireless/new-indoor-navigation-technologies-work-where-gps-cant>, [Accessed: 5 May 2017]
- [5] Anon 2017. 10 Heuristics for User Interface Design: Article by Jakob Nielsen [Online]. Available at: <http://www.nngroup.com/articles/ten-usability-heuristics/> [Accessed: 5 May 2017].
- [6] Anon 2017. About iBeacon on your iPhone, iPad, and iPod touch [Online]. Available at: http://support.apple.com/kb/HT6048?viewlocale=en_US&locale=en_US [Accessed: 5 May 2017].
- [7] Anon 2017. Apple push notification service. [Online]. Available at: <https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html> . [Accessed: 5 May 2017].
- [8] Anon 2017. Beacons | Google Developers [Online]. Available at: <https://developers.google.com/beacons/> [Accessed: 5 May 2017].
- [9] Anon 2017. Build a DIY iBeacon with a Raspberry Pi [Online]. Available at: <http://www.makeuseof.com/tag/build-diy-ibeacon-raspberry-pi/> [Accessed: 5 May 2017].
- [10] Anon 2017. Cite a Website - Cite This For Me [Online]. Available at: <https://image.slidesharecdn.com/ase-featuredrivenddevelopmentfinal-140826084449-phpapp01/95/feature-driven-development-13-638.jpg?cb=1409042792> [Accessed: 5 May 2017].
- [11] Anon 2017. Cite a Website - Cite This For Me [Online]. Available at: <https://image.slidesharecdn.com/ashortintroductiontoibeacons-140508112527-phpapp01/95/a-short-introduction-to-ibeacons-20-638.jpg> [Accessed: 5 May 2017].

- [12] Anon 2017. Cite a Website - Cite This For Me [Online]. Available at:
https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Art/model_view_controller_2x.png [Accessed: 5 May 2017].
- [13] Anon 2017. google/eddystone [Online]. Available at:
<https://github.com/google/eddystone> [Accessed: 5 May 2017].
- [14] Anon 2017. InMobiSYS '04: Proceedings of the 2nd international conference on mobile systems. *Bluetooth and WAP push-based location-aware mobile advertising system*. (InMobiSYS '04), pp. 49–58.
- [15] Anon 2017. Performance testing for DevOps | Load Impact [Online]. Available at:
<https://loadimpact.com/> [Accessed: 5 May 2017].
- [16] Anon 2017. What are region Monitoring and Ranging? [Online]. Available at:
<https://community.estimote.com/hc/en-us/articles/203356607-What-are-region-Monitoring-and-Ranging-> [Accessed: 5 May 2017].
- [17] Beacons, E., Beacons, L., Stickers, E. and Mirror, E. 2017. Estimote [Online]. Available at:
<http://estimote.com> [Accessed: 5 May 2017].
- [18] Industries, A. 2017. Adafruit Industries, Unique & fun DIY electronics and kits [Online]. Available at: <https://www.adafruit.com/> [Accessed: 5 May 2017].
- [19] Logic W. 2017. What really is the "business logic"? [Online]. Available at:
<https://softwareengineering.stackexchange.com/questions/234251/what-really-is-the-business-logic> [Accessed: 5 May 2017].
- [20] L. Aalto, N. G'othlin, J. Korhonen, and T. Ojala. Bluetooth and WAP push-based location-aware mobile advertising system. InMobiSYS '04: Proceedings of the 2nd international conference on mobile systems, applications, and services, pages 49–58, Boston, MA, 2004. ACM Press [Online; Accessed: 7 Feb 2017].
- [21] Kiran Thapa and Steven Case. An Indoor Positioning Service for Blue-tooth Ad Hoc Networks. Technical report, Department of Computer & Information Sciences, Minnesota State University, [Online; Accessed: 7 Feb 2017].

11. Appendices

The test results from the use case testing and functional and Interface testing can be seen in the appendix A.