KING'S College LONDON

University of London

6CCS3PRJ Final Year Individual Project Report Title

Final Project Report

Author: Sari Nusier

Supervisor: Dr. Andrew Coles

Student ID: 000000

April 12, 2016

Abstract

The abstract is a very brief summary of the report's contents. It should be about half-a-page long. Somebody unfamiliar with your project should have a good idea of what your work is about by reading the abstract alone.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Sari Nusier

April 12, 2016

Acknowledgements

It is usual to thank those individuals who have provided particularly useful assistance, technical or otherwise, during
your project. Your supervisor will obviously be pleased to be acknowledged as he or she will have invested quite a lot of time overseeing your progress.

Contents

Chapter 1

Introduction

This is one of the most important components of the report. It should begin with a clear statement of what the project is about so that the nature and scope of the project can be understood by a lay reader. It should summarise everything that you set out to achieve, provide a clear summary of the project's background and relevance to other work, and give pointers to the remaining sections of the report, which will contain the bulk of the technical material.

1.1 Report Structure

Chapter 2

Background

2.1 Global Positioning System

"GPS answers the questions 'What time, what position, and what velocity is it?' quickly, accurately, and inexpensively anywhere on the globe at any time" (Hofmann-Wellenhof, Lichtenegger and Collins, 1997).

In order to locate aircrafts and ships, the U.S. military developed a system called TRANSIT, the predecessor of the modern positioning system. It used six satellites at an altitude of 1,100 km, passing over one point on earth roughly every 90 minutes. The system worked by using the Doppler effect, the change of a wave's frequency for an observer relative to movement, to compute its location. This meant that users had to wait between "fixes" quite a long period, making the system unusable for real-time positioning, but enough for military applications (Hofmann-Wellenhof, Lichtenegger and Collins, 1997).

GPS solved this problem by ensuring that at least four satellites were always electronically visible. This allows for a receiver to compute its location using trilateration. The increase in accuracy and the ability to find oneâĂŹs location in real-time, made the GPS suitable for civilian applications.

Due to a higher complexity of indoor environments and the lack of line-of-sight transmissions between the receiver and the satellites, GPS is not suitable for indoor applications (Gu, Lo and Niemegeers, 2009).

2.2 Indoor Positioning System

"In the last years a great deal of research has been conducted on developing methods and technologies for automatic location-sensing of people or devices" [3].

Like the GPS, an IPS is a system that continuously and in real-time can determine the position of something or someone in a physical space [2]. Unlike GPS, IPS must available for indoor locations. Gu, Lo and Niemegeers [2] The first classification method they use is based on whether the systems uses the current infrastructure or not. A network-based approach will require no additional hardware, considerably reducing the cost. On the other hand, a non-network-based approach might offer higher degrees of accuracy, as the infrastructure must be designed and set-up for the specific application. Another classification method used is based on the medium used to determine location: Infrared (IR) is a common system because of the high availability of IR and its presence in everyday consumer devices. Roy et al. defines an Active Badge system which uses diffused IR to send signals every 15 seconds [3],[4]. Receivers are placed in each room, collecting the data and using it identify the location of the different emitters [3],[4]. This is a possible solution for the museum application, as the main focus of the system is to identify the room or section that the visitor is currently in, high position . This can be also applied to specific exhibitions, where receivers can be placed, reading the signal from the nearby phones. There are multiple downsides of this approach. Mainly, it is hard to read data that comes in parallel from multiple devices. This problem would appear if there are multiple devices sending a stream of bits simultaneously. Shortening the duration of each transmission can reduce the likelihood of collision, but will also drop the bitrate, reducing the number of unique badges that can be generated. Another downside is the lack of IR blasters in smartphones. Even though the Android SDK offers infrared capabilities, the official android compatibility guide does not make any reference to infrared, whereas bluetooth compatibility is mandatory [5]. The usage of radio is also very common in indoor positioning systems. These can range from using RFID tags to identify people as they enter rooms, to bluetooth transmitters, but also the use of WLAN networks that are already installed on sight. As mentioned above, using the already installed WLAN system represents a network-based approach, which is cost effective and time effective, whereas the non-network-based approach offers a greater deal of control [2]. This project will keep the goal of creating a positioning system which is relatively independent of the means used for measuring distance. But for convenience, it will use bluetooth modules to measure signal strength, and therefore, distance.

2.3 Machine Learning

"The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories". (Bishop, 2006) One part of machine learning is concerned with supervised learning, in which the training data is a set of input vectors, given along with their corresponding target vectors. When the goal is to associate each input vector to one of a finite set of categories, it is called a classification problem.

Chapter 3

Requirements and Specifiations

HOST represents the management of the building implementing the positioning system. VISITOR is any user of the positioning app, independent of the building they are visiting. Both users will require Wi-Fi enabled Android Devices and a stable internet connection.

3.1 Requirements

The requirements are split in four categories. The first two are concerned with user requirements from both HOST and VISITOR perspective. The positioning system is the software that handles data analysis and machine learning to provide the position of the VISITOR.

3.1.1 Host requirements

- H1 - Host must be able to create, edit, remove or view buildings.
- H2 - Host must be able to create, edit, remove or view rooms from specific buildings.
- H3 - Host must be able to measure WiFi APs signal strength for a specific room.
- H4 - Host must be able to update the positioning system with new measurements.
- H5 - Host must be able to reset the positioning system when needed.

3.1.2 Visitor requirements

- V1 - Manually select a building to visit.
- V2 - Receive a suggestion regarding the building they are in.

- V3 - Choose the rooms/exhibits they would like to visit inside a building.
- V4 - See an estimated time of visit based on selected rooms/exhibits.
- V5 - Rate the degree of importance for visiting a specific room/exhibit.
- V6 - See a suggested visit path based on selected rooms/exhibits.
- V7 - See its current location within the building.
- V8 - See a visit path based on his current location and the selected rooms/exhibits.

3.1.3 Positioning requirements

- PS1 - Receive a learning set in JSON containing AP signal strength measurements and the location they were taken in.
- PS2 - Create classifiers from learning set.
- PS3 - Receive unclassified data in JSON containing AP signal strength measurements.
- PS4 - Classify received data using one or more previously built classifiers.
- PS5 - Return classified data in a format that is easily parsable by the user

3.1.4 Backend requirements

- B1 - Database should store buildings and the rooms within each building.
- B2. Database should store learning set for each building.
- B3. Server should maintain a RESTful API to interface with the database and positioning system.

3.2 Specification

The specifications show what should be implemented in order to meet the requirements of the software.

3.2.1 Host

For requirements H1 to H5 an android application will be developed for the administration of the buildings created for a HOST. The app should contain:

- Activity to view all the buildings available (H1)
- Activity to add a new building (H1)
- Activity for the management of a specific building (H - 1, 2, 4, 5)
- Activity for the management of a specific room (H - 2, 3)
- Make HTTP requests to the Backend Server

3.2.2 Visitor

For requirements V1 to V8 an android application will be developed and should contain:

- Activity to vie all the buildings available (V1)
- Search through the building list by name (V1)
- Move the current building, if detected, to the top of the list (V2)
- An activity containing a list of exhibits/rooms inside a building (V3)
- Compute the sum of the estimated visit time of all the selected rooms (V4)
- Implement a slider for each selected exhibit (V5)
- Display as a text or as a path on the map (V6)
- Current location should be displayed as a text or on the map (V7)
- Update the suggested path based on current location (V8).

3.2.3 Positioning system

For requirements PS1 to PS5 a Java application will be developed and should contain:

- Accept TCP connections to send and receive JSON data. (PS - 1, 5)
- Implement machine learning and JSON parsing libraries. (PS - 1, 2, 3, 4, 5)

3.2.4 Backend

The backend should implement a database and an HTTP server to:

- Accept HTTP requests for the REST API
- Store data in JSON format.

3.2.5 Limitations

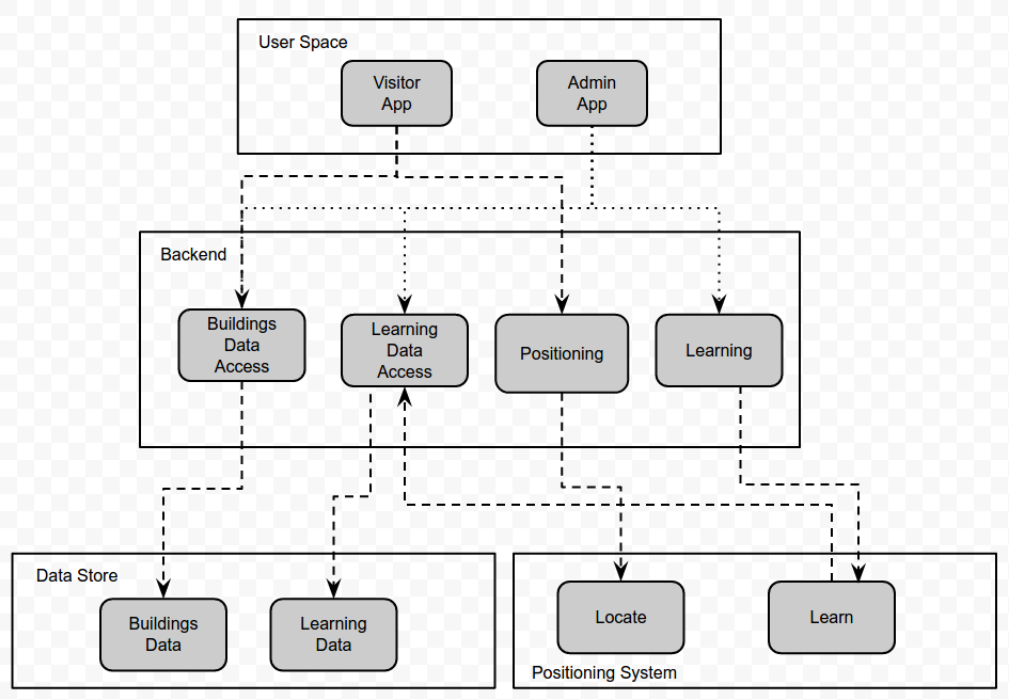
When developing the system, security was not a main consideration. The system does not implement any form of authentication or access control. It does not distinguish between multiple users, all data being commonly accessible and anonymous. The data is available only through the REST API, and any request which is not formatted accordingly will be ignored. Arbitrary code execution and any type of code injection has been considered and security measures implemented. The design of the backend is made to be extendable and any additional security measures can be easily implement. The user interface was not a priority, therefore the design is simple, but usable. Outputs are usually plain text, and visual cues are rarely given. The visual indoor maps and all the visual cues (location pin, walk path, etc.) are implemented using the Google Maps API, therefore, are available only for the buildings that have implemented Google Indoor Maps.

Chapter 4

Design

4.1 Architectural design

In this section we will define the architecture of the entire system, showing the dependencies between its components.



The architecture of the system is comprised of three subsystems. The top subsystem is concerned with interfacing the end users (HOST and VISITOR) to the lower subsystems. The middle subsystem offers an API to the bottom subsystems. The bottom subsystems manage and provide the information required for the entire system to run. To continue, we will review each subsystem in more detail, from top to bottom.

4.1.1 User Interface

This layer is comprised of two Android Application, one for each type of user.

The following are common features for both applications:

1. Implement serializable classes for the models defined in the database
2. Request and receive data from the backend in JSON format.
3. Post data to the Backend in JSON format.
4. Parse JSON data to create Java Objects
5. Convert Java Objects into JSON.
6. Activity to display all the buildings available in a ListView
7. Buildings ListView must be clickable and, on click should launch another activity for the specific building.
8. Clicking on a room should launch an activity for the selected room.
9. Implement Wi-Fi capabilities in order to measure signal strength of all visible APs.

These common features will be implemented in packages that can be shared across the two apps.

Administration application (HOST)

Due to the difference in requirements a separate android application must be developed for the HOST. In addition to the common features, the application will also implement the following:

1. A button to add a new building In addition to the list of available buildings.
2. An add building activity containing a form and a submit button for creating a new building
3. Selecting a building from the list opens an activity containing fields that can be used to edit the buildingâĂŹs information.
4. A clickable ListView with all rooms in a building as part of the activity mentioned above.
5. A ”learn” button to update the Positioning system with new data.
6. Clicking on an item in the Rooms ListView will open an activity that will allow the apps to measure the signal strength of all APs, associating them with the selected room, posting to the database.

7. A button in the Building activity to add new rooms.

8. The add room button will open an activity containing a form and a submit button for creating a new room.

Visitor application

In addition to the common features, the visitor application will also implement the following:

1. A clickable ListView with all the buildings available.

2. Selecting a Building will open an activity containing a list of all rooms as selectable items. Selecting an item will update the Estimated Time value displayed somewhere below.

3. A button to start the visit will open another activity that either displays the Google Indoor Map if available, or a textview.

4. The Visit Activity displays the current location of the visitor as a pin on the Map or as text.

5. The Visit Activity will also display a recommended path for the visit on the Map or as text.

4.1.2 Backend

For the simplicity of the software, and for a more efficient integration of all subsystems, the backend is ran using a NodeJS server, implementing a RESTful API. The API handles HTTP methods to "http://serveraddress/res/", where res is the resource targeted. In addition /res/id can be used, where available, to apply method to a resource that has the specified id. The resources available and the results of each method are as follows:

- For resource "buildings/":
  - GET: Returns a list of all buildings stored in the database
  - POST: Adds a new building to the database
- For resource "buildings/id/":
  - GET: Returns the building with the specified id
  - PUT: Updates the building with the specified id
  - DELETE: Deletes the building with the specified id

- For resource ”rooms/id/”:
  - GET: Returns the rooms that belong to the building with the specified id
  - POST: Adds a new room to the building with the specified id
  - DELETE: Deletes the room with the specified id
- For resource ”measurements/”:
  - GET: Returns all measurements taken on the system
  - DELETE: Deletes all measurements taken on the system
- For resource ”measurements/id”:
  - GET: Returns measurements taken for the room with the specified id
  - POST: Adds a new measurement associated to the room with the specified id
- For resource ”learn/id”:
  - GET: Updates the positioning system for the building with the specified id
- For resource ”locate/id”:
  - POST: Returns the predicted location in the building with the specified id

4.1.3 Positioning System

This is the most important part of the project. It is the software which implements the machine learning and data processing algorithms in order to create the classifiers needed for positioning. This subsystem is built out of a X modules, each one with its own task. It is implemented using the Weka Machine Learning libraries. The software implements the following features:

1. Opens TCP connections in order to receive requests and data.
2. A JSON Object is received containing the request type and the data to be used.
3. For a request of type ”learn” the software parses the data received and creates a classifier using it.
4. In order for this to happen, the software will require to convert the data into and ARFF file format.
5. A list of all the rooms is extracted and saved as a local file in JSON format. The file is named using the ”building_id”+”_buildings.json”

6. A list of all the Reference Points (RPs) is extracted and saved as a local file in JSON format. The file is named using the building_id + _RPs.json

7. Using the learning set ARFF file, a classifier is initialised and added to an array for future use.

8. For a request of Type "classify", the system checks whether a classifier has been built for the building id given in the request.

9. If a classifier has been previously built but it has not been initialised in the current run of the system, the server will initialise a classifier using the data saved from the previous learn command called for the building.

10. If a classifier has never been built, the system will return an error.

11. Once the classifier is initialised, it will be used to classify the data received in the request, and will return the room as a result.

12. The system implements multiple algorithms therefore a format must be implemented to return multiple results given by different algorithms. The data is returned as follows: [algorithm_name]:[predicted_room_id],[algorithm_name]:[predicted_room_id], etc. In other words, the algorithm name and the room predicted by it are separated by ":" and the algorithms are separated using ",".

Prediction System:

When designing the system, multiple methods for finding the location of the device were considered.

Trilateration

Firstly, dimensions of the building and its rooms, and the coordinates of each RP must be known. An RSS measurement is then taken for each visible RP: one next to the RP and one at the furthest corner from the RP, in the same room. To increase precision, more measurements at each of the two points can be taken and the results averaged.

We define:

MRP_0 - Average signal strength taken at the RP coordinates.

MRP - Average signal strength taken at the furthest corner from the RP, in the same room.

DRP - Distance (in meters) from the RP's location to where MRP was taken.

In perfect conditions the Distance as a function of Signal Strength increases linearly. Therefore,

to convert Signal Strength to Distance is a regression problem.

Using the equation of the line

$$y = mx + n$$

we must find the gradient m and the y-intercept n.

We compute

$$m = \frac{DRP}{MRP - MRP_0}$$

and

$$n = DRP - m * MRP$$

m and n values are then saved for each RP and the distance will be calculated as

$$Distance = M * RSS + N$$

To find the location of the device based on the distance measured we must find the intersection of n circles (where n is the number of RPs taken into account and must be greater than 3 if the coordinates must be found in 2 dimensional space, or 4 if in a 3 dimensional space). To determine the intersection points for 3 RPs we will use the algorithm defined by Paul Bourke [8]:

First calculate the distance d between the center of the circles $d = |P_1 - P_0|$

If $d > r_0 + r_1$ or $d < |r_0 - r_1|$ then there are no solutions. If $d = 0$ and $r_0 = r_1$ then are an infinite number of solutions.

Considering the two triangles $P_0P_2P_3$ and $P_1P_2P_3$ we can write

$$a^2 + h^2 = r_0^2$$

$$b^2 + h^2 = r_1^2$$

Using $d = a + b$ we can solve for a,

$$a = \frac{r_0^2 - r_1^2 + d^2}{2d}$$

It can be readily shown that this reduces to r_0 when the two circles touch at one point. Solve for h by substituting a into the first equation,

$$h^2 = r_0^2 - a^2$$

So

$$P_2 = P_0 + \frac{P_1 - P_0}{d}$$

And finally, $P_3 = (x_3, y_3)$ in terms of $P_0 = (x_0, y_0), P_1 = (x_1, y_1) and P_2 = (x_2, y_2)$ is

$$x_3 = \frac{x_2 + -h(y_1 - y_0)}{d}$$

$$y_3 = \frac{y_2 - +h(x_1 - x_0)}{d}$$

The problem with this approach is that it assumes that the distance measurements are precise. In our case, where the readings are not precise, errors can occur, such as one or more circle not intersecting in one exact point or, intersecting at all. This can be solved by finding the closest point to all circles, such that the sum of all distances from a point to the circle be the minimum.

Probability based approach

Because the specification of the software favours precision over accuracy, it is therefore a better approach to try and guess the room in which the device is found, instead of the actual coordinates. Therefore, an approach is to calculate the probability of a device to be in a room, based on statistic data.

Each room must store a list of average readings for all the reference points visible from that room. We define:

- $AVG(RP_x, R_y)$ as the average signal strength value for RP_x taken in room R_y.
- $N(R_y)$ as the number of RPs that can be measured from room R_y.
- mRP_x as a signal strength measurement for RP_x.

A reading is given as a list of mRP, that is, the readings taken at a moment in time. The goal is to use this list and calculate the probability that the readings were taken in a specific room.

For each room R_i, we calculate S for each measurement mRP_j, if $AVG(RP_j, R_i)$ exists and is

different than 0, as follows:

$$S(RP_j, R_i) = |1 - \frac{mRP_j}{AVG(RPj, Ri)}| + 1$$

Giving us a score from 1 to infinity.

We then calculate

$$\frac{\sum_{i=1}^{M} \frac{1}{S(RP_i, R_x)}}{N(R_x)}$$

where M is the number of measurements in the set. The result is the probability for the measurements to have been taken in room R_x. We can then compare the probability for all rooms and choose the one which is most likely.

Machine Learning approach

Machine Learning algorithms can be applied to this application in order to predict the room in which a device is located. We first define a learning set of all measurements taken and the rooms that the measurements were taken in.

There are five classifiers considered: Naive Bayes, Bayes Networks, KStar, Decorate and NB Trees. The classifiers are implemented using the Weka Machine Learning Library. The result of each classifier is returned to the backend, therefore, the decision of which predicted value to be used is moved higher up in the system. Due to a higher level of precision, the Machine Learning approach will have the priority in implementation.

4.1.4 DATA FLOW and STATE MACHINE

Because of the vast amount of data that must be transferred between the subsystems, it is important to visualise and understand what data must be inputted in each subsystem, what data is outputted by each subsystem and how this data moves around in the entire system.

The initial input is in the HOST android application. Here, the user inputs the details of a new building to be created. The data is then converted into JSON and sent to the Backend through a POST request to "/buildings". The backend parses the data and creates a new Building object, storing it in the database. The HOST then can select the building created and begin adding new room. For each new room, the details are entered and submitted. After submitting each room, the data, in JSON format, is sent to the Backend using a POST request to "/rooms/building_id". The backend then created a new Room object, setting its building id

to the specified id and storing it in the database. After the required rooms are added, the HOST must then take the RSS measurements for each room, in order for the classifier to be built. In the room activity, the HOST starts the measurements, and a list of all measurements taken at a point in time is then sent to the backend through a POST method to "measurements/room_id", where the backend stores it in the database. After enough measurements were taken for all the rooms, the host can then press the "learn" button from the building activity, which will send a GET request to "/locate/building_id". On receipt of this request, the backend queries the database for all the measurements taken in the building and sends it through a TCP connection to the Positioning system, as a "learn" request. The positioning system processes the request and data, parsing it and storing the data locally. It will then initialise a classifier using the data stored, the system is then in standby again.

The next trigger is by the VISITOR app, which, on start, will request a list of buildings from the backend using GET "/buildings/". The backend the queries and returns all the buildings stored. The user can then press on a building and see the list of rooms available. When the user starts a visit, the app starts collecting RSS measurements. For every scan, the list of measurements is sent to the backend using a POST request to "/locate/building_id". This triggers the backend to send a "classify" request to the backend server, which returns a list of predicted locations. The backend the forwards the information back to the user.

4.1.5 Data Store

The Data Store contains two types of data, Building related and Learning related. Building data is all data concerning the operation and visualisation of the buildings and the rooms contained. Learning data is the data used by the Positioning System. Initially, this layer was designed using an MySQL, but due to compatibility and performance issues with Android and longer development time, it has been changed to MongoDB.

The following data models were defined to be used with the database:

RECTANGLE:

- 4 coordinates (X,Y), one for each angle: Left-top (LT), Right-top (RT), Left-bottom(LB), Right-bottom (RB)

BUILDING:

- Name: The name of the building as a string.
- Rectangle: The rectangle model that defines the building.

- Width and Height(length)

ROOM:

- Name: The name of the room as a string.
- Rectangle: The rectangle model that defines the room.
- Width and Height(length)
- Floor: The floor number in which this room is located.
- Estimated Time of Visit
- Building ID: to associate each room to a building

Reference Point (RP): This is a point in space for which distance measurements are taken.

- RPID: A unique identifier
- Building ID: To associate each RP to a building.
- Coordinates (X, Y)

RP Measurement: Distance to an RP as measured in a specific room.

- RP - Value Pair: Contains the RPID and the value of the measurement
- Room ID: The ID of the room in which the measurement was taken

Chapter 5

Implementation

This chapter will go through the development process of the project. It will look into how the design was implemented, what changes have been made, what problems were faced along the way and how these problems were remediated.

5.1 Development approach

The project has taken a lean software development approach. The main goal was to achieve a prototype system, in order to test its feasibility. Therefore, some features were dropped or disregarded, as they were not crucial to the production of a testable system. Second principle of lean was also applied, analysing the value of the software based on its fitness to be used, rather than on the conformance to requirements. Therefore, focus from documentation or planning was moved towards coding, so that different ideas can be tried out and dropped if they proved unfeasible, or if they did not increase the quality of the software. Another lean principle applied was giving the system, as a whole, priority when developing each subsystem. Even though one of the objectives of this software is to create somewhat independent subsystems, so that they can be reused or changed easily later on, some compromises were made to ensure that the system as a whole was developed as soon as possible and was available for testing.
The following list shows, in order, the modules developed and the requirements targeted.

1. Even though not a priority, the Visitor app was the first software created. The initial android application was used to check whether the tools available through Google Indoor Maps API were usable for this project (V6, V7, V8).

2. Data models classes were created in Java, adding some of the basic attributes and func-

tionality (H1, H2, B1).

3. Later on, more models were added and the classes became serializable, to allow the to be sent (H1, H2, B1).

4. Data models for the positioning system were added (B2, H4).

5. Work on backend started. A MySQL database was created and java class to connect and query the database was implemented (B1, B2).

6. Tables were created in MySQL and Spatial Data Types were tested (B1, B2).

7. NodeJS server, MongoDB and basic data models were created for experimentation (B1, B2, B3).

8. MySQL database was dropped and data models were developed for MongoDB. REST API was also developed to allow manipulation of the newly created database (B1, B2, B3).

9. Administration app was created (H1- H5).

10. Java class for integrating with the API was implemented and added to the Administration app (H1-H5).

11. Buildings Activity was implemented to show the list of all the buildings available (H1).

12. Added building creation capability in the backend and in the ApplicationsâĂŹ Database helper class (B1, H1).

13. Building creating activity implemented in the Admin app (H1).

14. Delete buildings and add rooms capabilities implemented in the Admin app (H1, H2).

15. Positioning models implemented in the Backend and Database (B2).

16. Wi-Fi scanning and posting measurements to server functionality implemented (H3, H4).

17. Positioning models changed for better intersystem compatibility.

18. Basic Machine Learning functionality using Weka implemented and tested (PS2, PS4, PS5).

19. Positioning server implemented after successful tests (PS1 - PS5).

20. Added the Java Database helper class to the Visitor app (V1 - V8).

21. Implemented Buildings, Room activities and started displaying current location of the device in Visitor App (V1, V3, V4, V7).

The beginning of the implementation focused more on Backend, in order to analyse the feasibility of different approaches. Once a basic version of the backend was implemented, the administration app was the next step in the implementation process. This allowed for data to be inputted easier so that the rest of the system can be tested faster, when needed. The Backend and Database were tested using PostMan tool. Having Wi-Fi scanning capability early on helped understand the nature of the measurements, contributing to the design of the positioning system. The positioning system was implemented separately, as a Java program, and data was parsed and fed into the system manually, for testing. Once the Positioning system proved feasible, data inputtin became automated and the implementation moved its focus towards the Visitor app.

5.2 Backend

Implementation of the backend started using a MySQL database. For easier administration MySQL Workbench software was used. A test database was created with two Tables. The first table, Building, contained five attributes: Name (VARCHAR), Width (INT), Length (INT), ID (INT) and Rectangle of type Geometry. The Geometry type is part of the Spatial Data Types offered in MySQL and it helps model and manage geometrical shapes [1].

Due to compatibility issues, the implementation of the backend moved to using a NodeJS server with a MongoDB database, which is the current implementation. To ease implementation, Mongoose object modeling library was issued when creating the Database models [2]. The four models implemented and their corresponding schemas are shown below.

INSERT CODE!!!

```
{
    "rectangle": {
        "lt":{"x":0, "y":0},
        "rt":{"x":0, "y":0},
        "lb":{"x":0, "y":0},
        "rb":{"x":0, "y":0}
    },
    "name": "Room 3",
```

```
    "width": 21,
    "height": 21,
    "floor": 0
}
```

Chapter 6

Professional and Ethical Issues

Either in a seperate section or throughout the report demonstrate that you are aware of the **Code of Conduct & Code of Good Practice** issued by the British Computer Society and have applied their principles, where appropriate, as you carried out your project.

6.1 Section Heading

Chapter 7

Results/Evaluation

7.1 Software Testing

7.2 Section Heading

Chapter 8

Conclusion and Future Work

The project's conclusions should list the key things that have been learnt as a consequence of engaging in your project work. For example, "The use of overloading in C++ provides a very elegant mechanism for transparent parallelisation of sequential programs", or "The overheads of linear-time n-body algorithms makes them computationally less efficient than $O(n \log n)$ algorithms for systems with less than 100000 particles". Avoid tedious personal reflections like "I learned a lot about C++ programming...", or "Simulating colliding galaxies can be real fun...". It is common to finish the report by listing ways in which the project can be taken further. This might, for example, be a plan for turning a piece of software or hardware into a marketable product, or a set of ideas for possibly turning your project into an MPhil or PhD.

Appendix A

User Guide

A.1 Instructions

You must provide an adequate user guide for your software. The guide should provide easily understood instructions on how to use your software. A particularly useful approach is to treat the user guide as a walk-through of a typical session, or set of sessions, which collectively display all of the features of your package. Technical details of how the package works are rarely required. Keep the guide concise and simple. The extensive use of diagrams, illustrating the package in action, can often be particularly helpful. The user guide is sometimes included as a chapter in the main body of the report, but is often better included in an appendix to the main report.

Appendix B

Source Code

B.1 Instructions

Complete source code listings must be submitted as an appendix to the report. The project source codes are usually spread out over several files/units. You should try to help the reader to navigate through your source code by providing a "table of contents" (titles of these files/units and one line descriptions). The first page of the program listings folder must contain the following statement certifying the work as your own: "I verify that I am the sole author of the programs contained in this folder, except where explicitly stated to the contrary". Your (typed) signature and the date should follow this statement.

All work on programs must stop once the code is submitted to KEATS. You are required to keep safely several copies of this version of the program and you must use one of these copies in the project examination. Your examiners may ask to see the last-modified dates of your program files, and may ask you to demonstrate that the program files you use in the project examination are identical to the program files you have uploaded to KEATS. Any attempt to demonstrate code that is not included in your submitted source listings is an attempt to cheat; any such attempt will be reported to the KCL Misconduct Committee.

You may find it easier to firstly generate a PDF of your source code using a text editor and then merge it to the end of your report. There are many free tools available that allow you to merge PDF files.