University of Manchester
School of Computer Science

MSc in Computer Security

An iPhone Application for Providing *iBeacon*-based Services to Students

Progress Report

Author:
Mohammed Binsabbar

Supervisor:
Dr. Ning Zhang

May 9, 2014

Abstract

iBeacon is an extension to the location service offered by Apple in its devices. It allows iPhone applications to become location-aware. iBeacon is based on Bluetooth Low Energy (BLE) and provides a proximity location service for an indoor environment. Nowadays, mobile phones are equipped with advance technologies. One of these technologies is BLE. BLE consumes less energy from the device, hence, making it a good candidate to use for indoor positioning system.

Mobile phones can be used to enhance student's services. One of these services is taking the attendance. Taking attendance requires a location factor of the student. Hence, iBeacon can be used for this purpose.

The project is going to implement iBeacon-bases services for students. This includes implementing two applications. A backend server, which has the location information in its database, and it offers the services. An iPhone application which uses iBeacon signals to request services from the server. This report outlines the current progress in the project.

List of Figures

List of Tables

Contents

1 Introduction

Since the introduction of the first consumer mobile phone in 1980s by Motorola, there have been continuos developments in the technology used to make mobile phones [24]. Thanks to these developments, nowadays, mobile phones are built using very advance technologies. For example, they are equipped with fast CPUs, large RAMs and different wireless technologies. As a result, mobile phones have become capable of performing complex tasks. In the past, these tasks would require conventional computers.

This advance progress in mobile phones have attracted more consumers. Based on figures by Gartner Inc, the number of mobile phone owners are growing [26]. This increase in mobile phone sales has created new usage for mobile phones, which helps in increasing the person's productivity. There is a wide range of mobile applications for increasing productivity. These applications range from role-specific applications to general-purpose applications. For example, timetable and assignment organiser designed specifically for students, and "To Do" applications for general purpose and day to day use.

A context-aware application is an application which can adjust its behaviour and functionality based on the context inputs around it. For example, in a location-aware application, the application can offer certain services to the user based on the user's current location. iBeacon is an extension to the location service offered by Apple iOS devices. It allows iOS application to become aware of its location in an indoor environment via iBeacon signals. iBeacon is a bluetooth low energy payload, which carries location information related to its beacon. iOS applications can make use of iBeacon to create new experience for the users. [17]

Given the advance in mobile phone technology and the availability of iBeacon facility, it possible to enhance and improve some student services. For example, iBeacon-enabled mobile application can be developed to automatically take the student's attendance.

1.1 Project Aims and Goals

The aim of the project is to explore the potential use of iBeacon facility as an indoor positioning system in Kilburn building to provide student

services. The project aims at:

1. Investigating the advantages of using *iBeacon* to provide location-based services in an indoor environment.
2. Investigating what student services can be improved by using iBeacon.
3. Proposing a set of enhancements for those student services, that can be improved.
4. Proposing some new student services, which can take advantage of the IPS.
5. Investigating the limitations and difficulties of using iBeacon.

The goal of the project is to design and implement iBeacon-based indoor positioning system in Kilburn building using **Estimote** [4] beacons. The project is going to design and implement the backend server application, which offers the location-based services to its client. Moreover, the project will produce an iPhone application (student's client) that makes use the indoor positioning systems in Kilburn building to request services from the server. Lastly, the project will design and implement a login protocol, which makes using the client application secure and convenient for the student.

1.2 Report Outline

In section 2, a background research about old and current positioning systems for outdoor and indoor environments is discussed. This is followed by a brief background discussion about technologies, which have been used to track student's attendance. Then, password-based client authentication is described. Section 3 describes how the project will be carried out. It talks about the software development approach. The section includes a description of the system evaluation. Section 4 shows the current progress of the project. It focuses on the design and implementation of the login protocol. It also includes a brief discussion of the current server application and iPhone application implementations. Finally, the report is concluded.

2 Background Research

2.1 An Overview of Navigation System Services

As stated in 'Global Positioning Systems, Inertial Navigation, and Integration' book [32], "there are five different forms of navigation". These forms are; Pilotage, Dead Reckoning, Celestial Navigation, Radio Navigation, and Inertial Navigation. The Radio Navigation, which the project is related to, makes use of radio signals in order to provide location service. For an outdoor environment, the Global Positioning System (GPS) has been widely used for civilian purpose since the 1980s. Before the 1980s, GPS had a restricted use, and used mainly for military purposes. There have been some attempts to provide indoor positioning systems that make use of radio frequencies, such as bluetooth and wireless. [36]

2.1.1 Global Navigation Satellite System (GNSS)

GNSSs depend mainly on satellite signals. Currently, there are two GNSSs in use. The first one is the Global Positioning System (GPS), which was developed by U.S Department of Defence. There are about 28 satellites operating in an orbit around the earth. They transmit signals, which carry helpful information, to a receiver device on the earth in order to enable the receiver to locate itself. Global Orbiting Navigation Satellite System (GLONASS) is another GNSS. It uses different configuration from the one used in the GPS. It was developed by the Soviet Union, currently Russian Republic. [32]

Two more GNSSs are in developments now. Galileo is a new satellite based-navigation system, which is being developed by the European Commission's Galileo Single Task Force. One of Galileo project goals is to provide more accurate and precise position than GPS [32]. Galileo is expected to be fully operational in 2019 [18]. China as well is developing a Compass Global Navigation System (CGVS), which is an expansion of the BeiDou Navigation Satellite System. BeiDou has limited operations and coverage. The CGVS is expected to be deployed in 2020 [15].

Originally the GPS was designed for military purpose only. Later, it has become available for other services for civilian use. Nowadays, GPS has many different applications, which range from providing precise time synchronisation to enabling navigation in an open environment [16] [14].

Google Maps application for mobile phones is one example of using GPS for navigation.

There are some technical issues with the use of GNSS, which could be resulted from inaccurate calculations by the receiving device. These have been addressed by upgrading the used equipments and improving the algorithms. Despite these technical issues, there are non-technical limitations as well. In general, in places where satellite signals are obstructed, like in an underground or an indoor environment, the use of GNSS becomes useless. Hence, some recent efforts have been made to provide an indoor positioning system. [27]

2.1.2 Indoor Positioning System (IPS)

An indoor positioning system is a system, which implements the necessary infrastructure in order to provide indoor location services to enable objects to be located in an indoor environment. The system uses different technology from the ones used in GNSS to determine an object's location. However, there are some similarities in the way a location is calculated. Currently, there is no standardisation for the technology used in IPSs, and many various techniques and solutions have been proposed. [39]

The purpose of the IPS is to provide location service in places, where satellite signals are partially or completely blocked, especially inside buildings. The complexity of the indoor environment structure affects the transmission of the satellite signals, hence, resulting in multi-path effects [35] [39]. For example, the GPS can detect that an object is in a building, however, it cannot determine where exactly in the building. Google maps uses public wireless signals installed in malls to provide indoor navigations, like it did in the Mall of America, in Bloomington, Minnesota [36].

IPS has various applications, which include navigation, tracking and monitoring services. In large indoor environments, such as airports, museums, stadiums or malls, IPS can provide navigation service to the users to help them reach the place they want in the building. Furthermore, the use of IPS can improve existing services, or even provide new ones. For example, installing IPS in a museum allows smartphones to provide information services to the users. When a user is walking in a museum and passes by an artefact, his smartphone displays information related

to it [23]. Tracking valuable objects is another service that can be offered by IPS, for example, placing tags on medical equipments in hospitals to track them in order to prevent thefts [28].

In [28], Yanying Gu, Anthony Lo, and Ignas Niemegeers define three different types of location information, which are offered by the IPS. When the IPS is capable of determining the exact position of an object, in which the system operates, then it is said to provide the *Absolute Location Information* of the object. This type of information is usually needed for indoor navigation and tracking services, because these services need the exact location of the object to provide correct result.

The second type of information is the *Relative Position Information*. The IPS provides location information for different, but related parts of the target object. This type of IPS is usually applicable to motion tracking controllers for games.

The *Proximity Location Information*, which is offered by *iBeacon*, is the third type. The IPS is only capable of determining if an object is within a specific area, but not able to determine where exactly in this area. For example, the IPS can determine in which room in a building an object is, but cannot determine where exactly in the room the object is. This type of information is good for tracking equipments, or tracking children in a garden.

It all depends on the type of services being offered. Some services only require the *Proximity Location Information* in order to operate correctly. For example, for the previously mentioned museum example, it is enough to provide the *Proximity Location Information* for an application in order to return relative artefact information.

As with any navigation or tracking system, IPS endures security and privacy issues. This is not limited to IPS, but to all positioning systems in general, which can detect user's position at any given time. Some devices allow the user to opt-out and disable the location service. This will result in disabling some features in the device, like outdoor navigation. [33]

2.1.3 Technologies Used for Indoor Positioning System

There have been many different implementations for IPSs. Different technology is used for each implementation. In the early 1990s in Cambridge, AT&T developed the *Active Badge Location System* [40]. It used Infra Red

(IR) signals for determining object location. Each active badge has a unique number, which is broadcasted each 15 seconds. These signals get picked up by IR sensors, which are installed in a room. The sensors transfer the information to a master station via a central network. The master station processes the location information and displays it in a visual form to the user. Because of the dependancy on a connected network between the sensors and master station, active badge faced difficulties in buildings with large rooms, like lecture theatre. Moreover, sunlight and fluorescent light affect IR signals.

In nature, bats use ultrasound to navigate in the dark. This has inspired AT&T to develop a new IPS using ultrasound, which is called *Active Bat Location System* [41]. The idea is close to the previous system. An active bat device emits ultrasound signals that contain a 48-bits unique address. Ultrasonic signal receivers are installed in the room's ceiling in a square grid shape. Receivers are connected via a high speed network to a master board, which calculates the results from the receivers.

Mobile phones are not equipped with IR and Ultrasound technology. Hence, the two previously mentioned technologies require extra hardware to be carried out or tagged in the object. Moreover, they do not make use of already installed hardware, like Wi-Fi.

Mobile phones are commonly equipped with Radio Frequency (RF) technologies, specifically Bluetooth (IEEE 802.15.1) and Wi-Fi (IEEE 802.11). The same is true for conventional personal computers. Researchers have implemented IPSs which are based on RF. The main motivation is that there is no need to install addition hardware to what is already in mobile phones. Moreover, many offices these days have Wi-Fi access points.

Microsoft developed RADAR, an RF-based indoor positioning system [20]. Existing wireless LAN infrastructure is used. Triangulation technique is used to determine the tracking object location, based on signal strength (SS) and the signal-to-noise ratio (SNR) values. Each access point (AP) measures these values from the target object. RADAR system performs location calculations based on the data collected by 3 APs. Ekahau is a similar system [3]. However, it requires special RF tags to be mounted on target devices. APs measure the received signal strength indicator (RSSI) and forward the data to a processing location engine. It performs the location calculation and makes them available to the users. As mentioned earlier, Google has already used WLAN in malls to provide

IPS for its Maps application [36].

2.1.4 Bluetooth Low Energy and iBeacon

Bluetooth Low Energy (BLE) is an improvement implementation over the classic bluetooth (bluetooth V2.1 and V3). The main and major improvement is the lower power consumption. BLE consumes less power, which allows BLE chip to run on coin-size batteries for months or even years. Apple was first smartphone company to announce support for BLE in its iPhone Operating System (iOS) in 2011 [29]. Followed later by Android OS, Microsoft Windows 8 and Blackberry OS. Two years later, Apple announced iBeacon in its World Wide Developer Conference (WWDC) during the announcement of iOS 7. iBeacon is Apple implementation for an indoor positioning system, which provides proximity location service based on RSSI 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 Table 1.

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

Table 1: iBeacon signal address components.

In addition to address information, the API provides the developer with another two important information about the signal. These information are:

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

Lastly, iBeacon itself does not provide a complete IPS. It helps developers to build their own system, which is based on iBeacon. To build an IPS in a building using iBeacon, the building must have iBeacon emitters installed in the rooms. An iPhone application is designed to detect the iBeacon signals. If the location information are hardcoded in the application, then it can be retrieved without querying a backend server. Otherwise, the application needs to query the backend server, which holds the location information.

2.2 An Overview of Attendance Tracking

The project is going to propose a iBeacon-based solution for taking attendance at the University of Manchester. Hence, it is worth reviewing existing solutions.

At the beginning of a lecture, a paper with the name of the registered students is passed between the students. Each student marks his attendance by signing in front of his name. At the end of the lecture, the paper is returned to the lecturer. Later, either the lecturer or an appointed member of the university staff manually enters the students attendance in the system. This is the traditional method for tracking the attendance.

The process is not efficient. It is time consuming, because the process is done serially. Furthermore, the attendance record does not become available instantly. Moreover, the student's identity needs to be checked in order to make sure he is the one signing for his name. Otherwise, it is hard to prevent a student from taking his friends attendance. The process is also prone to errors. The whole process is done manually. It could be the case where a student does not get hold of the paper, then he forgets about it. Or the student's attendance is skipped unintentionally

during the manual input to the system. Lastly, the collected record does not provide information about attendance time.

Other forms for taking the attendance exist. Dicle and Levendis, from Loyola University New Orleans, have implemented an electronic attendance tracking system. It is based on **R**adio **F**requency **ID**entification (RFID). Each student is given an RFID tag and each lecturer is given RFID reader. The student scans the tag, when he enters the room. The attendance with the time is taken by the reader. The cost for a tag is 29 cents, and a reader is 20$. The reader is small, and is connected to a conventional PC. [22]

A similar system has been implemented by Northern Arizona University. The system makes use of the RFID chips, which are embedded in the student cards to monitor the attendance. A card reader is installed in the lecture theatre. The reader is sensitive and able to detect RFID tags round it. Hence, when student enters the class, the reader automatically records his attendance. The reader costed about 250$. This did not include the reader's installation and linking to the university's network. Another issue with this system is endorsing students privacy. The system is able to track the student's movements in the university, if the reader is installed everywhere in the campus. [34]

Some universities use the barcode on the student card to take the attendance in lectures [38]. It is also used by students to make use of library facilities. A barcode reader is used to read the student id off the barcode. Other universities, such as Texas State University, employs another approach, where student card is swiped through a card reader to log the attendance. The attendance record then becomes available online via web-based application [2].

2.3 Password-Based User Authentication

Entity authentication is the process of verifying the identity of an entity. There are four ways of which an entity can prove its identity. The four authentication methods are [37]:

- **Something the entity knows**: This form of knowledge could be a password, PIN or answer to personal questions.
- **Something the entity possesses**: This could be a shared secret key between two entity, electronic card, physical key, or a location.

- **Something the entity is (static biometric)**: Fingerprint and Retina.
- **Something the entity does (dynamic biometric)**: This could be voice pattern or typing rhythm.

In client-server architecture, the authentication can be one-way or mutual. In one-way authentication, only one of the two communicating parties is authenticated to the other. Whereas in mutual authentication, both entities are authenticated to each other. Password-based authentication have been widely used to authenticate the client. Since the introduction of digital certificate, it has been used for server authentication. The use of digital certificate plays a major role in the design of the Secure Socket Layer Protocol (SSL).

Password is the most common approach for client authentication. The main reason is that passwords are easy to remember. The client proves his knowledge of the password by sending it to the server. However, this approach is vulnerable and not used. Other entities might be on the same communication channel. Hence, the common approach is to prove knowledge of the password without sending it over the channel. An authenticator sends a string n to the client. The client proves his knowledge of the password by transforming n to a new value using function F with the password. The result value does not reveal the password, but it proves knowledge of it. This is commonly known as challenge-response authentication. There exists different implementation of this approach. For example, the implementation of function F is different. F can be an encryption function or a one-way function (hash function).

Two-factor authentication has recently been adapted by many online companies like Apple, Google, Dropbox and Github to improve password-based authentication. In the two-factor authentication, the password is used as 'something the user knows', where is the 2nd factor is used as 'something the user possesses'. It is a one-time code, usually supplied via SMS message to a registered token (a mobile phone). Google Authenticator app (for iOS, Android and Blackberry OS) allows the user to generate the codes, instead of receiving SMS messages [8]. Apple uses a different approach. It pushes the code via Apple Push Notification Service [1] to a registered iOS device [6].

2.4 iBeacon-based Student Services with Authentication Capability

2.4.1 The Origin of the Project Idea

In an undergraduate final year project last year, I have designed an iPhone application to improve student's productivity [21]. One of its functions is allowing students to take their attendance by scanning a QR code. The QR code is generated by the lecturer for each lecture. It is then displayed via a projector and the students scan it by the iPhone camera. The iPhone application analyses the QR code, then uploads the result to a server. There are obvious limitations with this approach. First, the lecturer needs to generate a QR code for each lecture. Secondly, it is hard to stop students from taking pictures of the QR code, and sending it to other students, who are not in the lecture theatre. This enables those students to take their attendance while they are not physically present. The location factor for taking the attendance was needed. Last year, iBeacon was not announced yet, and the only practical location factor was to use GPS. However, as discussed previously in **Section 2.1**, GPS is not good for indoor positioning. Moreover, IPSs developments were not active in the developers community.

2.4.2 The Requirements for Taking Attendance

Assuming a student is already enrolled in a course unit. For his attendance to be registered, he has to be present at the location and time on his timetable. This fact shows a very potential use of iBeacon for this purpose. An application can be designed to detect student's presence at a location using iBeacon. However, the problem is partially solved. Using iBeacon does not prevent one student from taking an attendance for another student using the same device. Hence, the mobile application must implement a mechanism, which prevents this from happening. To do this, the student must be authenticated to the server. As described earlier, password is the common approach for client authentication. It has also been discussed that other factors are used to authenticate a client. iBeacon can be used as a location-factor when authenticating the student, during the login process.

3 Research Methods

3.1 Software Development Methodology

One of the aims of the project is to investigate possible utilisation of iBeacon facility in the university to both improve or add new student services. During the investigation, new requirements might appear, and already existing requirements might disappear or change. However, by the end of the project, a prototype iPhone application will be produced, which demonstrates the possible utilisations. The process of developing the application is going to be iterative and the project will meet its requirements incrementally over the project iterations.

3.1.1 Iterative and Incremental Approach

The software will be developed by a signal developer. Hence, the project development is going to be '**User Story**' based incrementation and iteration. This means, each iteration focuses on only one single user story to be designed, implemented and tested. A user story is a descriptive sentence of one function that the system must provide. User story contains three important information, which are:

- **The Who**: The actor; the person, who is going to use the function.
- **The What**: The system requirement.
- **The Why**: The goal the function achieves when it is done.

For example, a possible user story of a system that allows students to login is; 'As a ***Student***, I want to be able to ***log in the system***, so that I can ***access the available services***'. However, the user story does not provide any details about the implementation or specifications. Moreover, the 'why' related part is optional.

3.1.2 Evolutionary Prototyping

One of the iteration goals is producing a working prototype of the selected user story. The final product is the final refinement of the original prototype. This is called evolutionary prototyping, where each iteration produces a working prototype which contributes to the final product.

When the requirements of the system are not fully defined, evolutionary prototyping approach becomes very helpful.

3.1.3 Iteration Length and Goals

Each iteration has goals to achieve by the end of its period. The iterations are short iterations, between one week to two weeks. The iteration period depends on the complexity of the chosen user story. The goals of each iteration are producing:

- Fully or partially working prototype of the selected user story.
- Bug Logs: It contains the bugs and issues encountered and should be fixed in the next iteration.
- Design Digram: The digram explains in high-level details the interactions and relationship between the entities and the system. The digram will expand as the software is being developed.
- Test package: The test package contains tests for this phase of development, and all the previous phases.

3.2 Defining the System Requirements

Due to the lack of real user and stockholder to meet with regularly, the requirements of the system have been defined by the developer of the application. There are 3 phases in which the requirements are gathered. These three phases are:

- **Brainstorming**: This is the initial stage for finding the requirements. The requirements are defined based on the developer's perspective.
- **Background Research**: The background research narrows the scope of the requirements that have been defined in the brainstorming stage. During this stage, existing solutions and limitations are explored.
- **Informal Meeting with Student**: These meetings are carried out informally with the students to reinforce the requirements that have been gathered in the previous two phases.

3.3 Software Design Pattern

The project involves designing and implementing both the backend server application and the iPhone application. One of the goals of the project is to provide a code that can be easily extended for future new functions at both ends. In order to achieve this, the project will follow common software design patterns.

Software design patterns define a set of patterns for the most common software design problems. A pattern is not a direct solution to the problem. It is however a guide to address the design problem. The design patterns help the software designer to make a reusable clean code. Moreover, it helps in minimising the necessary changes for the code when the requirements change.

In the book written by what is commonly known as the 'Gang of Four (GoF)', a set of 23 design patterns have been described [25]. They have been divided into three categories, which are:

- **Creational Patterns**: These patterns are concerned with the creation of an object and related objects. Examples of patterns in this category are Singleton and Abstract Factory.
- **Structural Patterns**: These patterns are concerned with defining structure for the relationship between objects. Examples of patterns in this category are Proxy and Adapter.
- **Behavioural Patterns**: These patterns are concerned with the communications between the objects and classes. Examples of patterns in this category are Observer and Visitor.

GRASP patterns are what Larman defines as the 'Fundamental Principles of Object Design' [31]. GRASP defines nine patterns. These patterns are: Low Coupling, High Cohesion, Information Expert, Creator, Controller, Indirection, Polymorphism, Protected Variations and Pure Fabrication. One or more of these principles fit in each of the GoF design patterns.

During the design of this project, it is aimed to follow the best practice in software design patterns and follow GoF and GRASP patterns where possible.

3.4 Out of Scope Issues

There are certain well known issues in the project. However, addressing these issues is not in the scope of the project. These issues are:

- **Smartphone Usage**: It is understood that not every student has a smartphone. Moreover, owners of smartphones are using different operating systems, such as Apple iOS, Google Android or Windows Microsoft. However, the project design aims at creating a system that works with any OS that supports bluetooth 4.0.
- **iBeacon Spoofing**: iBeacon signals can be emitted by any BLE emitter. Current iBeacon specification does not provide a method to authenticate the iBeacon signal. Because ProximityUUID, Major and Minor values are broadcasted publicly, then it is possible to forge an iBeacon emitter by using the same broadcasted values.

3.5 Selection of Tools

3.5.1 iBeacon Emitter

The selection of iBeacon emitter does not make a difference with regard to the iBeacon signal. Emitters will differ usually in signal range, device size, CPU and battery life. Estimote [4] was one of the first companies to announce iBeacon emitters. Estimote provides the developer with an easy to use API, which has been built on the top of Apple iBeacon API. Moreover, Estimote developers community is expanding. Estimote beacons are powered by a 32-bit ARM Cortext M0 CPU with a 256 KB flash. Figure 1 shows an Estimote beacon and its board.

3.5.2 Server Hosting

The design and implementation of the server infrastructure is not part of the project. However, the application that runs on the server is within the scope of the project. For this reason, an existing infrastructure solution has been chosen. Heroku is a hosting service for server application. Heroku allows the developer to solely focus on the design of the application rather than the underlying infrastructure. Heroku takes care of all the infrastructure design and security considerations. Heroku offers a set of servers to run the applications on. The choice of the server depends on

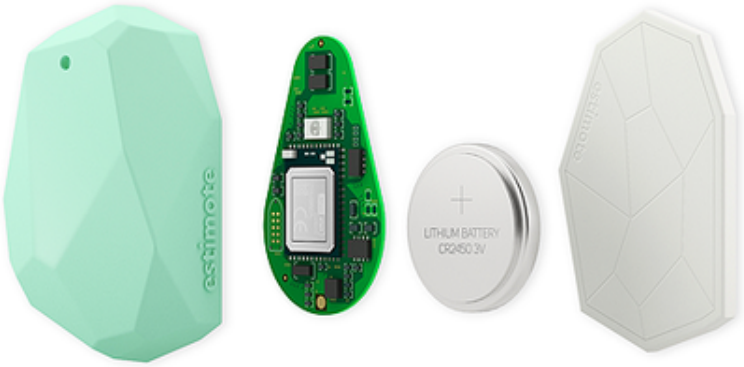


Figure 1: An Estimote beacon [19].

the chosen language, which the server application is written in. Heroku supports Java, Ruby, Python, Node.js, Scala, PHP, Clojure and Play.

The server application is going to be written in Python. The main reason for choosing Python is the desire to improve Python programming skills. The server is gunicorn, which is "*Python Web Server Gateway Interface (WSGI) HTTP Server for UNIX*" [7]. Flask [5] framework is used to write the application. Flask handles the communication with the underlying server, gunicorn. Flask provides the developer with HTTP response and request wrappers. Request routing and response construction are done via Flask API as well.

3.5.3 Application Development Environment

Currently, the only company that natively supports iBeacon for its mobile operating system is Apple. For this reason, iOS has been chosen to be the client application platform. iOS runs on iPhones, iPads and iPod touches. It is still possible to write iBeacon-based application for other operating systems as long as they support BLE. However, an iBeacon bluetooth profile has to be implemented by the application itself. This is to enable the OS to pickup iBeacon signals.

Objective-C is Apple's programming language for writing iOS and OS X (Apple's desktop platform) applications. xCode is Apple's IDE for

developing iOS and OS X applications. xCode is developed by Apple and available for free.

It is free to develop and test iOS application on an iOS simulator on the OS X. However, in order to test the application on a physical device or distribute the application in Apple App Store, it is required to join iOS Developer Program, which costs $99 a year.

3.6 System Evaluation

At the end of the development phase, two applications will be produced. The first application is the server application, and the second application is an iPhone application. The server application handles all the HTTP requests from the iPhone application to provide iBeacon-based services. On the other hand, the iPhone application is responsible for detecting iBeacon signals, and sending service requests to the server. The available services are the system requirements.

Both the server and the iPhone applications will include unit testing for each iteration. Moreover, for each implemented requirement, there is going to be a functional testing. The unit testing tests the internal functions of the applications individually. However, the functional testing tests the requirements the application promises to deliver. In other words, functional testings verifies if the application has satisfied its requirements or not. In addition to these test, a survey will be conducted with students to evaluate how beneficial the iBeacon-based services are. The survey will also include UI related questions as well.

The project includes a design of a login protocol. The protocol is going to be evaluated based on how hard it is to be abused, and how easy to use by the student.

Lastly, Estimote beacons are used as an iBeacon emitter. The project will evaluate how effective iBeacon signals are. It will test different placement for the beacons in a room. It will also evaluate the number of beacons needed in room to provide the best possible accurate measurement.

4 Project Progress

This section discusses in details the work that has been done since March 2014. It focuses on the design of a login protocol for the system.

4.1 Current System Requirements

The following are the current system requirements, which the project aims at implementing:

- **Logging in The System**: The student must be able to use the iPhone application to log in the system, so he can request a service.
- **Taking Attendace:** The student uses the iPhone application to take his attendance. The application detects his location by picking up iBeacon signals and sending them to the server. The server checks the student's timetable, and marks his attendance if he has a lecture at the location and time in the request.
- **Checking Room/Theatre Information:** The student uses his iPhone application to checks information related to the room he is in. The application picks up iBeacon signals and sends them to the server. The server looks up the address of the room and returns room's information.
- **Booking Room/Theatre:** The student uses his iPhone application to book a room he is in. The application sends the room's iBeacon address to the server, then the server books the room if it is available.

4.2 Designing Login Protocol

The system architecture is a client-server architecture, Figure 2. Student services are offered by the server and accessed via the internet by a mobile application. The internet is a dangerous place. Hence, security measurements must be considered. There are security requirements, that have to be satisfied by the system, to protect it from being abused. The user of the mobile application is a student. The student has to log in to request location-based service from the server.

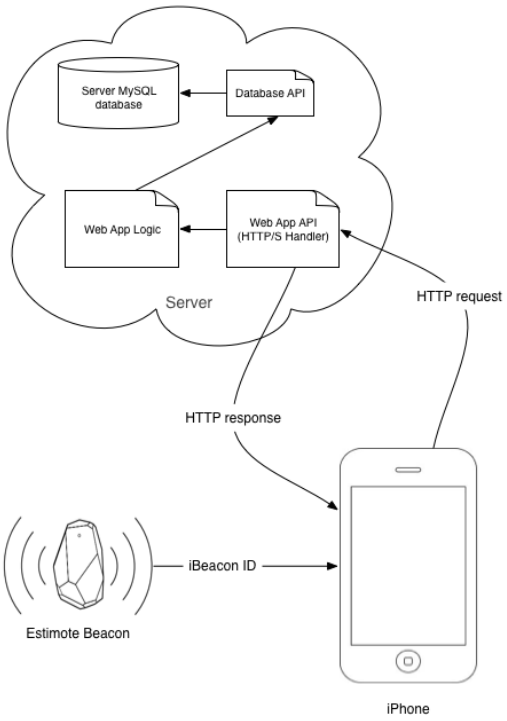


Figure 2: The system architecture

4.2.1 Security Requirements

The following are 7 security requirements for logging in the system. The word 'client' is used to mean the 'mobile application'.

1. The server must be able to authenticate the student.
2. The server must only accept requests, that are originated from the client.
3. The server should prevent a student from using two different devices to log in the system in the same time frame.
4. The server should prevent more than one student from logging in using the same device at the same time frame.
5. The client must be able to authenticate the server.
6. The use of the student's password should be minimised.
7. The use of the network bandwidth should be efficient. That means, minimising the number of messages between the client and server, and the use of public key cryptography.

4.2.2 Assumptions

There are some assumptions that must be taken into account, which are:

1. The student is registered with the university and is already able to use the university's wifi service.
2. The university enforces a strict password policy to ensure that a student does not choose weak passwords, for example, short and common words.
3. The server has a table with all the registered students. It contains student IDs and hashed values of the students' passwords.
4. Each student's password is hashed with a different salt value. Both the hashed password and the salt are stored in the server's database in different tables.
5. The server has a valid certificate. It is pinned in the iPhone application. Hence, it is possible to create a trusted self signed certificate by the server.
6. The student does not have an access to the application source code.
7. The server keeps record of students, who have unexpired service pass. Each student ID has either 1 or 0 entry in the service pass table. An entry in the table means the student has an active service pass. Otherwise, it means he does not an active session. Having active service pass allows him to request services without using his password.
8. The Estimote iBeacons are installed in secure places in the building. Hence, it is not possible to illegally change their places. Moreover, changing the address of an emitter is not possible without having admin privilege.
9. iBeacon signals are only broadcasted in a building, and the signal cannot be picked up from outside. Moreover, access to the building is guarded and requires a valid student ID.

4.3 The Login Protocol

The *Login Protocol* is divided into two protocols. The first protocol is used when the client does not have a *ServicePass*. The second one is performed

when the client has an unexpired *ServicePass*, and wants to request a service.

4.3.1 Abbreviations

This is a table with the abbreviations used in describing the protocol.

Abbreviation	**Description**
U	Student ID number, which is used as the username.
PWD_U	The password associated with username U.
Salt_U	A salt value used to hash the password of username U.
HashedPWD_U	A hashed password of U.
C	The client, which is a mobile application.
DUID	Device Unique Identifier, which uniquely identifies a device. This string cannot be modified or changed.
S	The server.
K_CS	A short term shared secret key between C and S.
AppKey	A key which is hardcoded in the client source code. It is used in hashing operation. This key is used by the client to authenticate origin of the request.
IP	An IP address of C.
Beacon_ADD	iBeacon emitter address, which is picked up by C.
N	A nonce.
HashedN	A hashed value of N.
ServiceData	This is the data for the requested service. It contains service type and other information.
TS	Timestamp.
LT	Lifetime.
||	Concatenation.
ServicePass	U || DUID || K_CS || LT
X —-> Y	Message sent from entity X to entity Y.

Table 2: Abbreviations used in the protocol description.

4.3.2 Requesting a *ServicePass*

The client performs the following three steps before performing the protocol with the server:

- Asks the student for his student ID (*U*) and password (*PWD_C*). This is the only interaction between the student and the client.
- Initiates an SSL session with the server. During this step, the client authenticates the server by verifying the server certificate.
- Starts an SSL connection with the server.

Once the above steps have been performed, the login protocol starts:

1. **At the Client Side:**
   - 1.1 Finds *DUID*. This is client platform dependent. Each client platform uses different approach to find this value. The value is always consistent, even if the device is rebooted.
   - 1.2 Detects two different iBeacon signals, that are the closest to it.
   - 1.3 Finds its IP address.
   - 1.4 Constructs a request and sends it to the server:
     C ⟶ S: *U* || *DUID* || *IP* || *Beacon_ADD* || *TS1*
2. **At the Server Side:**
   - 2.1 Validates the *IP* of the incoming request. The IP address is verified by checking that it is coming from the University of Manchester network address.
   - 2.2 Validates *Beacon_ADD* in the request. The is done by checking the existence of the addresses in the server database. Moreover, the two addresses must belong to the same location (same room/theatre/area).
   - 2.3 Checks the existence of *U* by querying the server database.
   - 2.4 Checks its database for a valid *ServicePass* that is associated with *DUID*. If there is a valid pass for the given *DUID*, then the associated student ID, *U'*, must be equal to *U*.
   - 2.5 If there is no valid *ServicePass* associated with *DUID*, then the server ensures that there is no other *ServicePass'* for *U* with *DUID'*, where *DUID'* != *DUID*.

2.6 If all the above validations pass, the server generates a nonce *N1*, finds the salt value of *U*, constructs a response and sends it to the client:
S —–> C: *N1* || *Salt_U* || *TS2*

3. **At the Client Side:**

3.1 Concatenates *PWD_U* with *Salt_U* and hashes the result to produce *HashedPWD_U*.

3.2 Concatenates *HashedPWD_U* with *N1* and HMAC with *AppKey* to compute *HashedN1*. *AppKey* is hardcoded in the application source code.

3.3 Constructs a request and sends it to the server:
C —–> S: *HashedN1* || *TS3*

4. **At the Server Side:**

4.1 Finds the stored *HashedPWD_U'* in its database.

4.2 Finds its version of *AppKey'*.

4.3 Computes *HashedN1'* by HMAC(*AppKey'*, *N1* || *HashedPWD_U'*).

4.4 If *HashedN1'* == *HashedN1*, the student is authenticated. Otherwise, login error is returned to the client.

4.5 If there is unexpired *ServicePass*, the server uses it. Otherwise, the server generates *K_CS*, and associates it with *U* and *DUID*. The server specifies *LT* for *K_CS*. These informations are inserted in the service pass table in the server's database.

4.6 Sends *ServicePass* to the client:
S —–> C: *ServicePass* || *TS4*

The login protocol is done after the client receives *ServicePass*. The client uses *ServicePass* for all the subsequent service requests to authenticate the student to the server.

4.3.3 Requesting Service

This protocol is performed by the client, as long as it has obtained a *ServicePass*. The following communication happens in insecure channel. Moreover, it does not require any student interaction, apart from selecting a service to request.

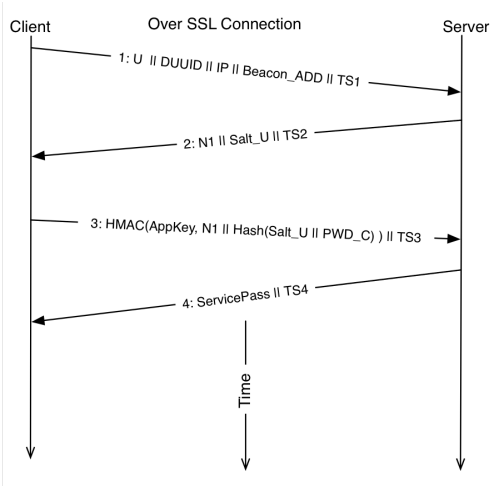


Figure 3: Summary of requesting a ServicePass protocol.

1. **At the Client Side:**

   1.1 Finds *ServicePass*, and extracts *K_CS* and *U*.

   1.2 Generates a nonce, *N2*.

   1.3 Constructs a request and sends it to the server:
   C —–> S: *N2* || *U* || *TS5*

2. **At the Server Side:**

   2.1 Finds *K_CS'* that is associated with *U* in its service pass table in the database.

   2.2 Computes *HashedN2* by HMAC *N2* with the key *K_CS'*.

   2.3 Generates a nonce, *N3*.

   2.4 Sends the following response to the client
   S —–> C: *HashedN2* || *N3* || *TS6*

3. **At the Client Side:**

   3.1 Computes *HashedN2'* by HMAC *N2* with the key *K_CS*.

   3.2 Checks if *HashedN2'* == *HashedN2*. Only the server knows *K_CS*. Hence, being able to produce the same hash value indicates knowledge of *K_CS*. This authenticates the server.

3.3 Finds *DUID* and concatenates it with *N3* and *AppKey*. It then HMAC the result using *K_CS*. *HashedN3* = HMAC(*K_CS*, *N3* ||*DUID* || *AppKey*).

3.4 The client constructs a request and sends it to the server:
C ⟶ S: *HashedN3* || *U* || *ServiceData* || *TS7*

4. **At the Server Side:**

4.1 Finds *DUID'* that is associated with *U* in the service pass table in the database.

4.2 Finds its own version of *AppKey'*.

4.3 Concatenates *N3* with *DUID'* and *AppKey'*, then HMAC the result using *K_CS'*. *HashedN3'* = HMAC(*K_CS'*, *N3* ||*DUID'* || *AppKey'*).

4.4 Checks if *HashedN3* == *HashedN3'*. This authenticates the user, verifies the used device and authenticates the origin of request.

4.5 Forwards the data in *ServiceData* to its handler to serve the client.

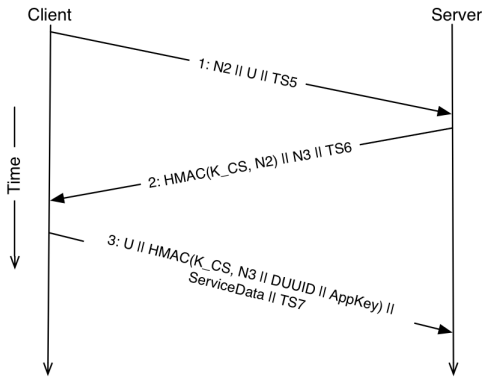


Figure 4: Summary of requesting a service protocol.

4.4 Security Analysis of the Protocol

The protocol satisfies all the security requirements in **Section 4.2.1**. In requesting *ServicePass* protocol, the client authenticates the student to the server using three factors, which are the student's password, IP and iBeacon addresses. On the other hand, the server authenticates itself using

its certificate during the SSL handshake. The client has a valid server certificate pinned in the source code. Pinning certificate allows the server to use a self-signed certificate. When the server needs to update its certificate, the client source code needs to be updated and pinned with the new certificate. In the second protocol, a mutual authentication between the server and the client is solely based on the knowledge of *K_CS*. Both ***Requirements 1*** and ***5*** are satisfied.

To attack *PWD_U*, *Salt_U* must be obtained from the server. To obtain *Salt_U*, the attacker must connect to the university wifi and pick up two different iBeacon signals of the same location. Only authorised university's users can connect to its WiFi network. Moreover, based on ***Assumption 11***, an attacker will not be able to pickup iBeacon signal, unless he is a student, or he uses a stolen student ID. Once the attacker obtains *Salt_U*, he can try to attack *PWD_U*. The difficulty of attacking *PWD_U* depends on its strength. Enforcing certain password policies helps in hardening password attacks. Even if the attacker obtains *Salt_U* to compute *HashedPWD_U*, he needs to know if the computed value is the right one. This means knowledge of *Salt_U* does not help without the correct *PWD_U*, and he must try to log in until he succeeds. However, the system suspends user access after 5 incorrect login attempts.

AppKey is used by the client to authenticate the origin of the request. This is to satisfy ***Requirement 2***. Given ***Assumption 6***, extracting *AppKey* requires decompiling the binary file, then analysing the assembly code. For the project case, the client is an iPhone application. Obtaining the source code requires jailbreaking the iPhone, which can be done by a naive user. iOS decompiler generates only the header files and assembly codes. Hence, knowledge of assembly code is required. Debugging millions of lines of assembly code is a hard task and requires a very motivated and experienced attacker. There are some techniques to harden debugging the decompiled code, such as using meaningless variable/method names instead of obvious names.

The protocol records the logged in devices by associating *DUID* with each valid *ServicePass*. *DUID* is consistence for each device. All the installed application in once device share the same value. However, to abuse the use of *DUID*, the source code is required to patch, and the difficult of this task is described previously. Moreover, because the server authenticates the origin of a request, then *DUID* must have been extracted

by the client, and has not been altered. The use of *DUID* is to satisfy both ***Requirements 3*** and ***4***.

Requirement 6 is satisfied by the use of *ServicePass*. The student uses his password only one time a day. The suggested lifetime of *ServicePass* is 8 hours. Given the fact the student has lectures from 9:00 to 17:00, then 8 hours is a reasonable choice. Moreover, the use of *ServicePass* requires only use of secure hash algorithm, hence satisfying ***Requirement 7***.

The use of nonce in the protocol prevents replay attacks. It enables both the server and client to ensure the request/response is fresh. When the client generates nonce, it is keet for a short period of time, 2-3 second. If the response does not arrive within the specified time, the client discards the nonce, so when it receives the nonce later, it refuses it. The server does exactly the same thing.

4.5 Selection of Hashing Algorithms

The protocol only makes use of hashing algorithms in its operations. It is always advised to use tested and existing solutions rather than inventing new ones. Existing and recommended solutions have been designed and tested by security expertise. Currently, there are no practical attacks on Secure Hash Algorithm with 256 bits output (SHA-256). Theoretical attacks exist, however, current computing power makes them impractical. [30]

SHA-256 is used for all the hashing purposes for the protocol. The length of *AppKey* must be 256 bits in this case, as it is used as a key for HMAC in the first protocol. Moreover, *K_CS* must be 256 bits long, because the mutual authentication in the second protocol is based on using it as a key for the HMAC. *AppKey*, *K_CS* and nonces are generated using secure random number generator. Each programming language provides its implementation for generating secure random keys. In the project case, Python is used at the server side, and it provides a function in its ***random*** package to generate secure random keys [11]. At the client side, Apple provides a function in its Security framework [12].

4.6 Current Implementation

The main goal of the first iteration was to setup the development environment and install the required third party frameworks.

4.6.1 The Server Application

The server application implementation focused on implementing the first login protocol. The current implementation of the protocol in the application is not complete. For example, the current implementation does not check for the IP address of incoming request. Also, currently all the passwords were hashed with the same salt value. Moreover, the current implementation was tested locally, and has not been pushed to Heroku servers yet.

The current application accepts only HTTPS 'PUT' request for logging in the system. The data format the application accepts is JSON. Because HTTP is stateless protocol, the server uses short life sessions in order to save the state of last HTTP request the server receives from the client.

4.6.2 The iPhone Application

The iPhone implementation has been smoother and faster due to using iPhone IDE. In the server application implementation, no IDE was used. Hence managing and organising files added extra time.

The protocol requires use of unique device identifier that is always consistent. However, Apple has deprecated the method, that returns the iPhone unique identifier. This value is guaranteed to be unique for every iPhone Apple makes. The reason for deprecating the method was to protect the user privacy. Apple provides an alternative solution, which returns a vendor identifier. However, as described by Apple "*The value of this property is the same for apps that come from the same vendor running on the same device. A different value is returned for apps on the same device that come from different vendors, and for apps on different devices regardless of vendor*" [13]. Moreover, the value changes when all the vendor's apps are deleted and one of them is reinstalled later.

OpenUDID [10] is an open source framework, and it aims at addressing the previously mentioned problem. The value generated by

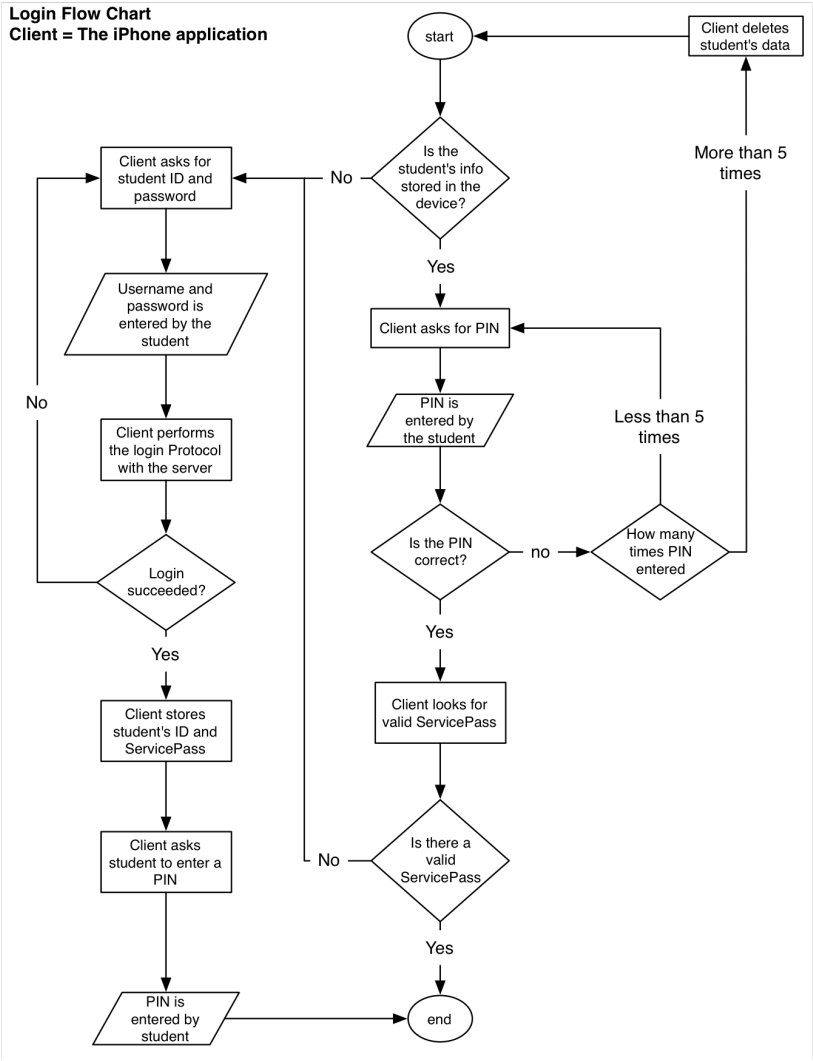
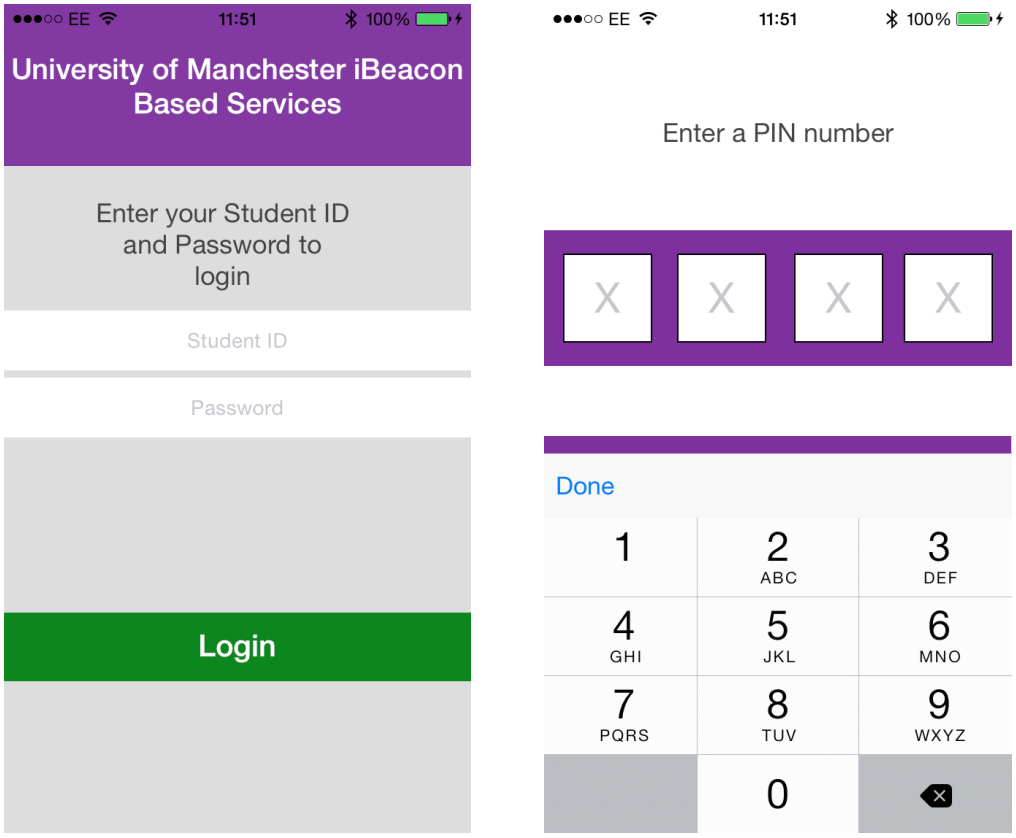


Figure 5: The iPhone application flowchart for logging in

OpenUDID is always consistent across the iPhone. Even if the user reinstalls the application, or restarts the device, the same UDID is generated.

Lastly, the iPhone application uses Apple Keychain Service [9] to save *ServicePass*. Keychain enables iPhone applications to store credentials securely in the device. Items stored via Keychain are encrypted when the iPhone is locked (The user has set a PIN or a password for the lock screen). The application flowchart for logging in is shown in Figure 5. Screenshots of the main login and PIN setup UIs are shown in Figure 6.



(a) The main Login UI

(b) PIN setup UI

Figure 6: Screenshots of the current UI for the iPhone implementation

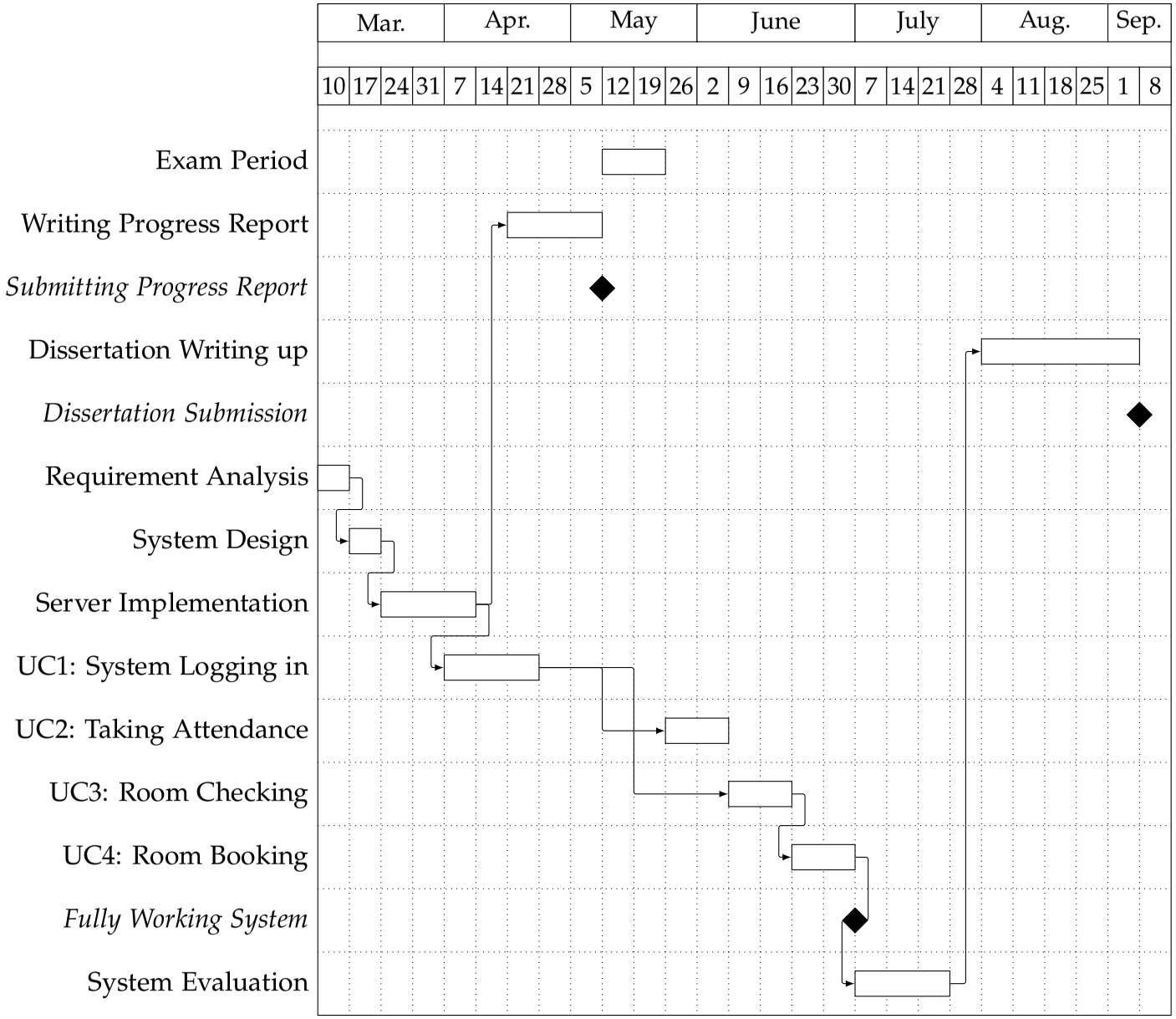
5 Conclusion and Future Work

The prototype produced in the previous iteration is going to be refined to fix some of the bugs. The main focus for the coming iteration is to fix the bugs and produce a prototype that fully implements the protocol at both ends. Once this is done, the project will focus on implementing the location-based student services. The coming iterations will implement these requirements in this order; *Take Attendance*, *Checking Room Availability* and *Booking a Room*. Some new requirements might appear, however, the project will focus on perfectly implementing the aforementioned three requirements. Each iteration is going to be a maximum of two weeks. Figure 7 shows the project plan as Gantt Chart. The Gantt Chart timeline is divided as weeks. The numbers in the 2nd row show the start date of the week.

iBeacon is a new IPS implemented by Apple. It makes mobile devices with BLE support to become location-aware. iBeacon does not require setup at the mobile device, apart from defining iBeacon profile. Apple devices already support iBeacon. Because iBeacon is BLE, it does not consume huge energy from the device.

A mobile application can be built to provide iBeacon-based services to the students. The application can make attendance taking an easy task. Moreover, the application can allow a student to a book a room, that he is in, if it is available. The student does not have to know the name of room. The application makes use of iBeacon to detect the student current location, and request a service from a backend server. The system can be expanded in the future to support more functions, such as navigation in the building.

Figure 7: Project Plan



References

[1] Apple push notification service. [Online; Accessed: 30 March 2014. Available at: https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html.

[2] Attendance tracking. [Online; Accessed: 28 Feb 2014. Available at: http://www.its.txstate.edu/departments/classroom_technology/attendancetracking.html.

[3] Ekahau; rfid-over-wi-fi tracking systems, rtls and wlan site survey. [Online; Accessed: 15 Feb 2014. Available at: http://www.ekahau.com.

[4] Estimote. [Online; Accessed: 17 March 2014. Available at: http://estimote.com].

[5] Flask. [Online; Accessed: 1 April 2014. Available at: http://flask.pocoo.org].

[6] Frequently asked questions about two-step verification for apple id. [Online; Accessed: 30 March 2014. Available at: http://support.apple.com/kb/ht5570.

[7] Gunicorn. [Online; Accessed: 1 April 2014. Available at: http://gunicorn.org].

[8] Install google authenticator. [Online; Accessed: 30 March 2014. Available at: https://support.google.com/accounts/answer/1066447?hl=en.

[9] Keychain services concepts. [Online; Accessed: 16 April 2014. Available at: https://developer.apple.com/library/mac/documentation/security/conceptual/keychainServConcepts/02concepts/concepts.html].

[10] OpenUDID. [Online; Accessed: 3 March 2014. Available at: https://github.com/ylechelle/OpenUDID].

[11] Python, random - generates pseudo-random numbers. [Online; Accessed: 10 April 2014. Available at: https://docs.python.org/2/library/random.html#random.SystemRandom].

[12] Randomization services reference. [Online; Accessed: 16 April 2014. Available at: https://developer.apple.com/library/ios/documentation/security/reference/randomizationreference/Reference/reference.html].

[13] UIDevice Class Reference: identifierForVendor. [Online; Accessed: 16 April 2014. Available at: https://developer.apple.com/library/ios/documentation/uikit/reference/UIDevice_Class/Reference/UIDevice.html#//apple_ref/occ/instp/UIDevice/identifierForVendor].

[14] GPS.gov: Road & Highway Applications. http://www.gps.gov/applications/roads/, 2006. [Online; Accessed: 7 Feb 2014].

[15] BBC News Technology: China's Beidou GPS-substitute opens to public in Asia. http://www.bbc.co.uk/news/technology-20852150, December 2012. [Online; Accessed: 7 Feb 2014].

[16] GPS.gov: Timing Applications. http://www.gps.gov/applications/timing/, September 2013. [Online; Accessed: 7 Feb 2014].

[17] iOS: Understanding iBeacon. Article, Apple Inc., http://support.apple.com/kb/HT6048?viewlocale=en_US&locale=en_US, 2013. [Online; Accessed: 28 Jan 2014].

[18] Galileo: Satellite launches. http://ec.europa.eu/enterprise/policies/satnav/galileo/satellite-launches/index_en.htm, January 2014. [Online; Accessed: 7 Feb 2014].

[19] Preorder for estimote beacons available, shipping this summer. Article, Estimote, July 2014. [Online; Accessed: 28 Jan 2014. Available at: http://blog.estimote.com/post/57087851702/preorder-for-estimote-beacons-available-shipping-this].

[20] P. Bahl and V.N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual*

Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 2, pages 775–784 vol.2, 2000.

[21] Mohammed Binsabbar. iStudentPocket: An iOS Application for The School of Computer Science to Organise Students and Provide Them with Easy Access to Their Academic Records. Final Year Project Report, School of Computer Science, University of Manchester, April 2013.

[22] Mehmet F. Dicle and John Levendis. Using RFID technology to track attendance. *Economic Educators*, 13(1):29–38, 2013.

[23] Jay Donovan. Wifarer Brings Indoor Navigation To The Royal BC Museum. `http://techcrunch.com/2012/08/01/wifarer-brings-indoor-navigation-to-the-royal-bc-museum/`, August 2012. [Online; Accessed: 10 Feb 2014].

[24] Tom Farley. Mobile telephone history. *Telektronikk*, 101(3/4):22–34, 2005.

[25] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[26] Gartner, Inc. Gartner Says Smartphone Sales Grew 46.5 Percent in Second Quarter of 2013 and Exceeded Feature Phone Sales for First Time. Press Release, August 2013. `http://www.gartner.com/newsroom/id/2573415`.

[27] Larry Greenemeier. A Positioning System That Goes Where GPS Can't. *Scientific American*, January 2008.

[28] Yanying Gu, Anthony Lo, and Ignas Niemegeers. A Survey of Indoor Positioning Systems for Wireless Personal Networks. *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, 11(1), 2009.

[29] Natalie Harrison and Teresa Brewer. Apple Launches iPhone 4S, iOS 5 and iCloud. online, October 2011. Available at: `https://www.apple.com/pr/library/2011/10/04Apple-Launches-iPhone-4S-iOS-5-iCloud.html`.

[30] Mario Lamberger and Florian Mendel. Higher-order differential attack on reduced sha-256. *IACR Cryptology ePrint Archive*, 2011:37, 2011.

[31] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Prentice Hall PTR, 2004.

[32] S. Grewal Mohinder, R. Weill Lawrence, and P. Andrews Angus. *Global Positioning Systems, Inertial Navigation, and Integration*. John Wiley & Sons, 2 edition, 2007.

[33] Ginger Myles, Adrian Friday, and Nigel Davies. Preserving Privacy in Environments with Location-Based Applications. *Pervasive Computing, IEEE*, 2(1):56 – 64, 2003.

[34] Mary Catherine O'Connor. Northern Arizona University to use existing rfid student cards for attendance tracking. *RFID journal*, 2010. [Online; Accessed: 28 Feb 2014. Available at: `http://www.rfidjournal.com/articles/view?7628`].

[35] P.Veeranath, Dr.D.N.Rao, Dr.Srinivasn Vathsal, and Bhasker Nalaveli. Reducing Multipath Effects in Indoor Channel for Analysis of GPS/Pseudolite Signal Acquisition. *International Journal of Scientific and Research Publications*, 3(2), February 2013.

[36] David Schneider. New Indoor Navigation Technologies Work Where GPS Can't. `http://spectrum.ieee.org/telecom/wireless/new-indoor-navigation-technologies-work-where-gps-cant`, November 2013. [Online; Accessed: 7 Feb 2014].

[37] William Stallings. User authentication. In *CRYPTOGRAPHY AND NETWORK SECURITY PRINCIPLES AND PRACTICE*, chapter 15. Prentice Hall, 5th edition, 2011.

[38] Hema Subramaniam, Marina Hassan, and Setyawan Widyarto. Bar Code Scanner Based Student Attendance System (SAS). *TICOM*, 1(3):173–177, January 2013.

[39] Kiran Thapa and Steven Case. *An Indoor Positioning Service for Bluetooth Ad Hoc Networks*. Technical report, Department of Computer & Information Sciences, Minnesota State University, Mankato, 2003.

[40] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.

[41] Andy Ward, A. Jones, and A. Hopper. A new location technique for the active office. *Personal Communications, IEEE*, 4(5):42–47, Oct 1997.