IPBeacons at Campus – Mobile Application

Shailab Chapagain - a40534
Susan Babu Pandey - a40528

Relatório Final do Trabalho de Projecto de Engenharia Informática apresentado à Escola Superior de Tecnologia e de Gestão do Instituto Politécnico de Bragança.

Trabalho orientado por:

Prof. Pedro Filipe Fernandes Oliveira

Prof. Paulo Matos

Bragança

2020-2021

IPBeacons at Campus – Mobile Application

Shailab Chapagain - a40534
Susan Babu Pandey - a40528

Relatório Final do Trabalho de Projecto de Engenharia Informática apresentado à Escola Superior de Tecnologia e de Gestão do Instituto Politécnico de Bragança.

Trabalho orientado por:

Prof. Pedro Filipe Fernandes Oliveira

Prof. Paulo Matos

Bragança

2020-2021

A Escola Superior de Tecnologia e de Gestão não se responsabiliza pelas opiniões expressas neste relatório.

Certifico que li este relatório e que, na minha opinião, é adequado no seu conteúdo e forma como demonstrador do trabalho desenvolvido no âmbito da UC de Projecto.

_____

Prof. Pedro Filipe Fernandes Oliveira (Orientador)

Certifico que li este relatório e que, na minha opinião, é adequado no seu conteúdo e forma como demonstrador do trabalho desenvolvido no âmbito da UC de Projecto.

_____

Prof. Paulo Matos (Co-orientador)

Aceite para avaliação da UC de Projecto.

Dedication

We dedicate this research project to our supervisor's and all the teachers for their great support even when things were so tough. They constanty kept on encouraging us to work extra hard, our friends and professors for moral support and encouragement throughout studies and for creating an enabling environment to carry out this project.

Acknowledgment

First and foremost, We would like to thank our project supervisor Prof. Pedro Filipe Fernandes Oliveira and Prof. Paulo Matos who guided us in doing this project. They provided us with advice, ideas and helped us . Those weekly conversation with prof. Pedro in order to discuss project related stuffs was really helpful and motivating. Their motivation and help contributed tremendously to the successful completion of the project. Besides, we would like to thank all the teachers who helped us by giving us advice and providing the equipment(Beacon).

Also, We would like to thank our family and friends for their support. Without that support we couldn't have succeeded in completing this project. At last but not in least, we would like to thank everyone who helped and motivated us to work on this project

Abstract

Whenever people move from place to place its vital for them to receive the certain information automatically from the particular place like campus, library, cinema, shopping market, etc. During this marketing era with a growing population it has been difficult to work in things like advertising, notifying, indicating, tracking and many other.

The purpose of this project is to implement a solution that allows the sending of information through personalized notifications to the user, when he approaches the different campus locations without any user interaction, using Bluetooth Low Energy (BLE) devices known as beacons. In order to do this, a mobile application called IPBeacon, that helps user to access and receive information, is intended to develop. In addition to this, we have developed the web page for the crud operation of beacons and its related things.

Taking the advantage of Beacon technology our goal is to develop an application to notify interest information to the user. We have collected the information of many types of beacons of today's market and analysed them, which will help us while configuring and developing application which will meet the objective we intended.

Key-words: Android, BLE, Beacon, IPBeacon, Mobile Application

Resumo

Sempre que as pessoas se deslocam de um lugar para outro, é vital que recebam certas informações automaticamente de um lugar específico, como campus, biblioteca, cinema, mercado de compras, etc. Durante esta era de marketing com uma população crescente, tem sido difícil trabalhar em coisas como a publicidade , notificação, indicação, rastreamento e muitos outros.

O objetivo deste projeto é implementar uma solução que permita o envio de informações por meio de notificações personalizadas ao utilizador, quando ele se aproxima dos diferentes locais do campus sem qualquer interação do utilizador, utilizando dispositivos BLE conhecidos como beacon. Para isso, o objetivo é desenvolver um aplicativo móvel chamado IPBeacon, que ajuda o utilizador a acessar e receber informações. Além disso, desenvolvemos a página da web para a operação de back-end dos beacons e coisas relacionadas.

Aproveitando as vantagens da tecnologia Beacon, nosso objetivo é desenvolver um aplicativo para notificar sobre informações de interesse ao utilizador. Coletamos as informações de vários tipos de beacons do mercado atual e a analisamos. O que nos ajudou na configuração e desenvolvimento de aplicativos que atendam aos objetivos definidos.

Palavras-chave: Android, BLE, Beacon, IPBeacon, aplicativo móvel

Contents

List of Tables

List of Figures

Acronyms

API Application Programming Interface. xx, 11, 21, 40, 50

BLE Bluetooth Low Energy. xi, xii, xx, 5–9, 13, 53

CRUD Create,Read,Update,Delete. xx, 27–29

EID Ephemeral Identifiers. xx, 10, 11

GHz Gigahertz. xx, 5

IDE Integrated Development Environment. xx

IOS iPhone Operating System. xx, 6, 9, 21

IoT Internet of things. xx, 15

IPB Instituto Politécnico de Bragança. xx, 1, 2

LE Low Energy. xx, 5

OS Operating System. xx, 6, 15

PDU Professional Development Unit. xx, 13

RSSI Received Signal Strength Indicator. xx, 9, 13

SDK Software Development Kit. xx, 21

TLM Telemetry Frame. xx, 10

TX Transmission Power. xx, 7, 9

URL Uniform Resource Locator. xx, 10, 12

UUID Universally Unique Identifier. xx, 7, 8, 10, 11, 13, 15, 22, 33

Chapter 1

Introduction

Daily movements of human beings, attached with the attractiveness of the increased automation of daily tasks, also comes in the context of daily mobility existing on a university campus, creating the need to automatically notify the user with interesting information requires the use of beacon technology. People travel from place to place and it won't be practical every time to connect to the network and search for the information or to provide information. Nowadays, we are experiencing huge population growth and rapid increment of retail business and marketing where advertising, tracking and indicating are the key things, so the value of beacon technology will also be increasing by time.

1.1 Framework

Our project is based on beacon technology. We are developing a mobile application named as **IPBeacon at Campus**. The application that we are developing interact with different beacons which are presented at different locations on Instituto Politécnico de Bragança (IPB) Campuses and show relative and customized information to the users.

Our application is intended to send the information through personalized notifications to the user, when he/she approaches the different campus locations, namely canteen, office, classrooms, bar, etc., in an non-invasive way and without any user interaction,

using beacon devices developed on the different spaces.

This application is mainly helpful for all users (Teachers/Students) who move around the campus. This application guides users for the easier movements around our campus, through personalized notification about the classes, canteens and etc.

To use our application for the very first time, users are expected to register in the application with their Username, email, password and number by which they are registered in IPB. After successful registration and login, they are redirected to the user's profile page and after that user can navigate to main interface, also called landing page, where they can view the various information which the app automatically generates, but information are visible only after the application detects beacons. The only thing user need to do is to turn on the Bluetooth and location, once it is enabled beacon is detected.

1.2 Objective

The main objective of our project is to develop and implement a solution that allows the sending of information through customized notifications to the user, when he approaches the different campus locations, namely canteen, library, classrooms, bar, etc., in an non-invasive way and without any user interaction, using beacon devices.

Our goal of the project is to reduce the effort of the user by the means of digitalized technology and to guide or help one easily by providing them vital information of various location without requiring them to explore anything from internet.
During this process, we had to learn, study, understand and explore beacon technology, and mobile application development. To achieve our goal and objective, the following working methodology was decided:

- Study of the existing platforms.
- Identification and analysis of the different types of existing beacon devices with its specification.
- Development of mobile application and its configuration with beacon technology.

- Learning and experimenting in real context.

1.3 Document Structure

This document is structured in the following chapters, so that the report can demonstrate all the work done in a systematic way.

Chapter 1 : Introduction

This chapter has the introductory approach to the project with a detailed description about why we develop the project.

Chapter 2 : Context and Technologies

This chapter is about the context of the work we are going to develop and the technologies we are using for the development of the project.

Chapter 3 : Approach Modeling

This chapter is about the system architecture and requirement analysis. Various models used for the development of the project are also presented here

Chapter 4 : Development

This chapter is about the implementation of the mobile application, how the application of the project is developed with the libraries used.

Chapter 5 : Tests Discussion

This chapter describes all the tests carried out and the discussions about the difficulties we faced during the preparation of our project.

Chapter 6 : Conclusion

This chapter has the conclusions about the work.

Chapter 2

Context and Technologies

In this chapter, we have covered the concepts of beacons along with their review in the market. This section also includes the tools used for the development and implementation for the practical part.

2.1 BLE

To understand about the beacons first we need to understand what **Bluetooth Low Energy (BLE)** is.BLE was introduced under the name Wibree by Nokia in 2006 and is part of the Bluetooth 4.0 specification since it was merged to Bluetooth standard in 2010. BLE uses different protocol than Classic Bluetooth and therefore is not backwards-compatible. Bluetooth LE uses the same 2.4 Gigahertz (GHz) radio frequencies.

The biggest advantage of BLE is its low energy consumption. This enables Beacons to transmit a signal continuously on a single button cell battery for a couple of years. As with the Classic Bluetooth, Low Energy (LE) can reach up to 100 meters.

2.2 BLE Beacons Protocols

BLE [1] beacons can be basically classified on the basis of protocol, power solution and location technology. The main protocols of BLE beacons that are available in the market

are :

- iBeacon (Apple)[1]
- EddyStone (Google)[1]
- AltBeacon (Radius Networks)[2]

2.2.1 iBeacon(Apple)

An iBeacon is a protocol/firmware launched by Apple in 2013[3]. This is the communication protocols which is most natively supported in iPhone Operating System (IOS) and has deep integration with the mobile Operating System (OS). IBeacon has served as the reference for the existing protocols. Although this technology was generally built by Apple, which is most natively supported in IOS, is also supported by Android, BlackBerry devices and with Windows Phone support (likely to arrive soon).

At figure 2.1, we can see a example of an iBeacon device

Figure 2.1: Example of iBeacon device[4]

How does iBeacon work ?

iBeacon uses BLE technology to transmit a unique ID that can be recognized by a mobile app or operating system on a smartphone. This unique ID is used to determine the smartphone's physical location and thus trigger a location-based action on the device.

As we know iBeacon uses a BLE Technology and BLE consists primarily of "Advertisements", or small packets of data, that broadcast at a regular interval by Beacons or other BLE enabled devices via radio waves. There are various data packets that a beacon(ibeacon) sends which is upto 31 bytes. Following are the lists of those packets[5] and fig 2.2 is the reference image to the same.

- iBeacon prefix
- Universally Unique Identifier (UUID)
- major number
- minor number
- Transmission Power (TX) power

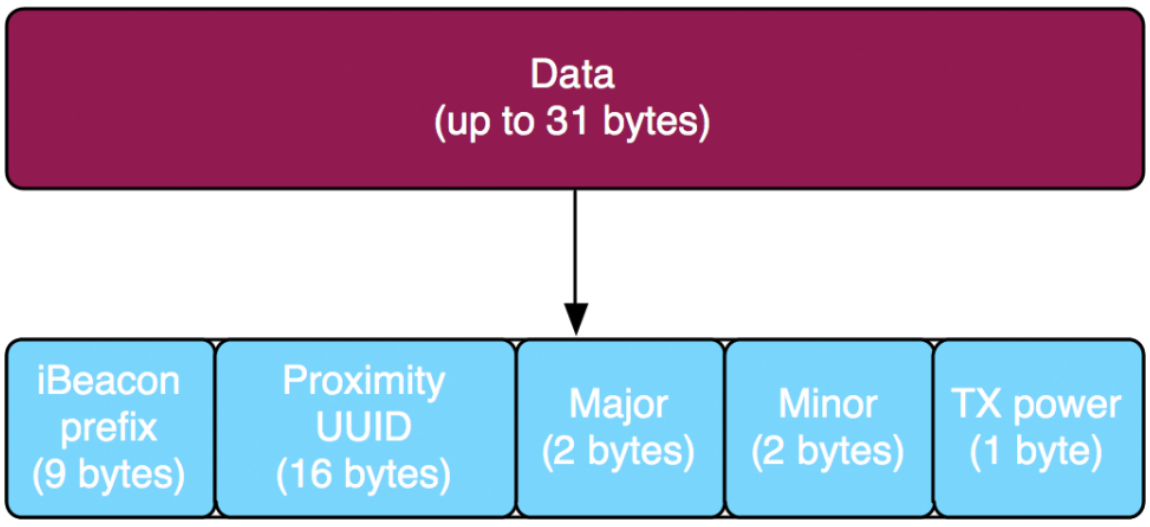


Figure 2.2: Ibeacon Packets[6]

The above blocks highlighted in blue are also divided into sub-blocks [7].
Here is the example in Appendix B

iBeacon Prefix(1st Line)

This is a fixed field, which contains manufacturer specific data, and also information about length of packet. iBeacon prefix actually indicates that this BLE beacon is actually an **iBeacon device**.
Appendix B contains an example.

UUID(2nd Line)

UUID is an identifier which is in the format **B9407F30-F5F8-466E-AFF9-25556B57FE6D**. This is commonly used to distinguish the beacons from others used in a particular context. Mobile app is then set up to listen just to this proximity UUID.

These are basically the 128-bit numbers that are assigned by the manager of the iBeacons to identify groups that will share functions.

iBeacons that are receive from the factory may be programmed with a default UUID that is specific to the manufacturer. Estimote beacons, for example, come programmed with a UUID of **`B9407F30-F5F8-466EAFF9-25556B57FE6D`**, Radius Beacons default to **`2F234454-CF6D-4A0F-ADF2-F4911BA9FFA6`**, and Kontakt.io beacons default to **`F7826DA6-4FA2-4E98-8024-BC5B71E0893E`** [1].

Major Number(3rd Line)

This is used to group a related set of iBeacons. What this means is that this number groups beacons into smaller partitions. Major values/numbers are basically intended to identify and distinguish a group. For example all beacons on a certain floor or room in a venue could be assigned a unique Major value.

These are unsigned integer values between **0** and **65535**

Minor Number(4th Line)

This number is basically used to identify a individual beacon among a group of beacons. For example distinguishing specific beacons within a group of beacons assigned a Major

value.

These are also unsigned integer values between **0** and **65535**.

TX Power(5th Line)

TX Power is also known as measured power value. This is used for distance estimation. TX power is the strength of the signal measured at 1 meter from the iBeacon. This number is then used to determine how close you are to the iBeacon. This value can be calibrated by the developer, and be equal to the average Received Signal Strength Indicator (RSSI) picked up one meter from the beacon. Usually the most accurate value for this field is set as default by the manufacturer.

Advantages of IBeacon

- This type of protocol is widely supported;
- This protocol is simple and easy to implement;
- IBeacon gives reliable performance on IOS;

2.2.2 EddyStone (Google)

Eddystone is a BLE format developed by Google and designed with transparency and robustness in mind. It's open and multiplatform, so you can use it with both Android and IOS. The protocol is now known for being "open", created with the input and collaboration of several companies [1].

How does Eddystone work?

How Eddystone works is the same as iBeacon. But there's extended functionality beyond that.An Eddystone Bluetooth beacon can broadcast four beacon packet types. These are also referred to as "frames" in the Eddystone Protocol Specification[1]

- Eddystone-UID(Unique ID);

- Eddystone-Uniform Resource Locator (URL)(Telemetry);
- Eddystone-Telemetry Frame (TLM);
- Eddystone-Ephemeral Identifiers (EID);

Eddystone-UID(Unique ID)

Of the three types of Eddystone frame , this is the one that directly corresponds to the functionality of Apple's iBeacon 1.0 packet. It's used to trigger actions in mobile apps that are running in the foreground or background. While the format is similar to iBeacon, the details of its structure differ sufficiently that the two are not compatible. An iBeacon device can't trigger an Eddystone app and a beacon that is only broadcasting to the Eddystone specification can't trigger an iBeacon app. Beacons need to broadcast both iBeacon and Eddystone-UID frames if they are to trigger actions in both iBeacon and Eddystone apps.

Eddystone-URL(Telemetry)

This is an alternative transmission to the Eddystone-UID that sends out a compressed 17 byte URL instead of a numeric identifier. The idea is that an app detecting the beacon can go directly to this URL without the app having to convert a beacon numeric identifier to destination web address. This Eddystone frame is the new replacement for the existing UriBeacon, an open standard also sponsored by Google.

Eddystone-TLM

Eddystone-TLM, as in 'telemetry'. This packet is broadcast alongside the Eddystone-UUID or Eddystone-URL packets and contains beacon's 'health status' (e.g., battery life). This is mainly intended for fleet management, and because of that, the TLM 'service' packet is broadcast less frequently than the 'data' packets.

Eddystone-EID

Eddystone-EID is a format similar to Eddystone-UUID, but with a single 8 byte AES-encrypted identifier that rotates every few minutes, hours or days depending on configuration. A beacon transmitting EID must be registered with Google's Proximity Beacon Application Programming Interface (API) using a registered project under a Google account.

Devices detecting an EID transmission can tell an Eddystone-EID transmitter is nearby, but cannot make sense of its identifier without credentials for the Google account with which it is registered. To make sense of a transmission, Google's Proximity Beacon API must be called to fetch usable information based on the ephemeral identifier. This adds an additional level of security for certain applications that may not want other parties to trigger app responses with their own beacons, or reuse beacon transmissions for their own purposes.

An example ephemeral identifier looks like this: 0a194f562c97d2ea, or like this with a prefix indicating it is a hexadecimal number: 0x0a194f562c97d2ea

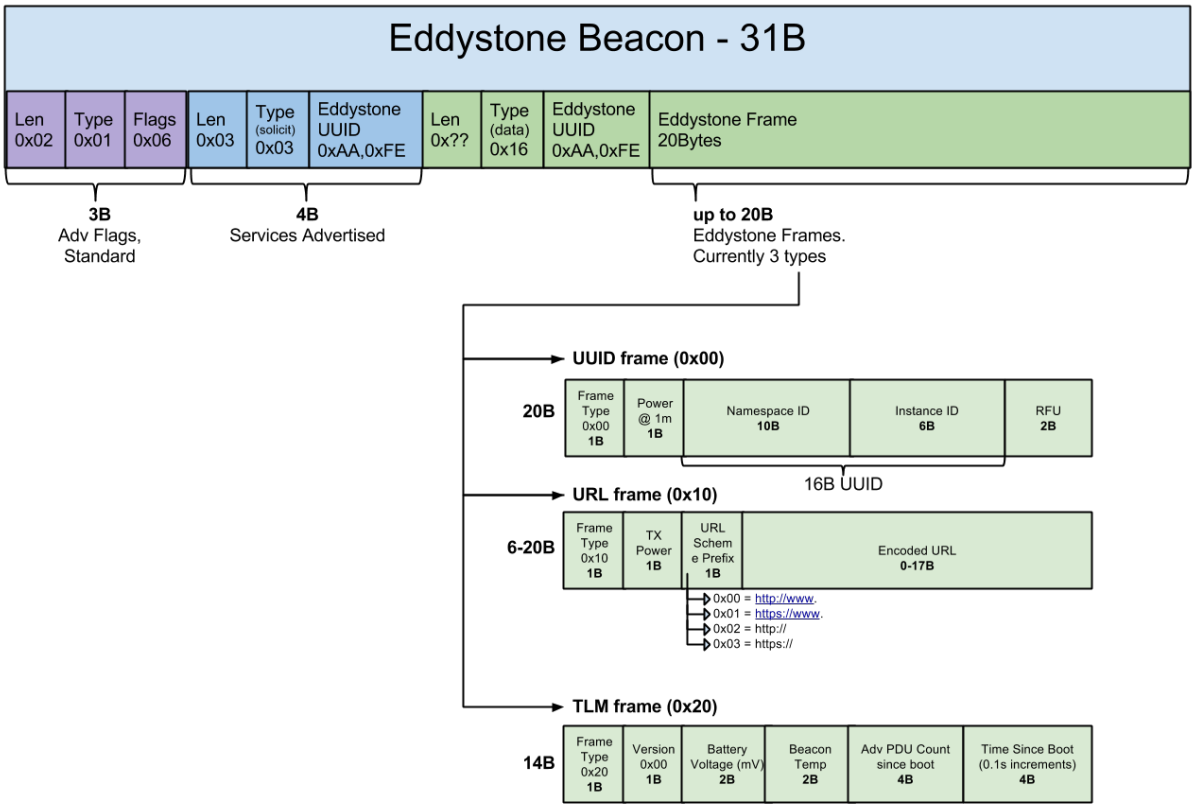


Figure 2.3: Eddystone Format [8]

Advantages of Eddystone

- This type of protocol can also send out URL, which removes the necessity of installed App, and Telemetry information;
- This protocol is open format and is known for its flexibility;

2.2.3 AltBeacon (Radius Networks)

AltBeacon, was designed by Radius Networks. It is free to use and anyone can implement beacons using their technology. Alt Beacons have specially been designed to create an open market for beacon applications [2].

The main feature of this beacon protocol is that it does not favor one vendor over

another for any reason other than the technical standards compliance of a vendor's implementation. AltBeacon uses a partially similar broadcast packet as the iBeacon so it can be easily integrated with it. The broadcast packages are divided.

Development Objectives

AltBeacon Specification development[9], here are the objectives :

- Provide a concise proximity advertising message for interchange of proximity information between advertisers and scanners;
- Maintain compliance with Bluetooth Specification Version 4.0 by utilizing defined advertising Professional Development Unit (PDU) and advertising data structures;
- Encourage adoption by all interested parties by avoiding any obvious implementation restrictions;
- Enable the implementation of vendor-specific features, if possible.

How does AltBeacon work?

The AltBeacon part is a 28 Byte long, of which 26 Bytes can be modified by the user. The first two bytes of the AltBeacon are set by the BLE stack, and can not be modified. ADV Length is 0x1B and ADV Type is 0xFF; these will specify the length of the advertising data packet and the type as manufacturing data respectively. The rest of the packet can be changed by the user. By default, it uses a UUID, a secondary ID (similar to the major), a third ID (similar to the minor), and a RSSI for calibration measured at 1m. AltBeacon includes after this an extra field of data of one byte that can be used for different purposes, for example to indicate battery level or temperature, but ultimately is the manufacturer's choice to implement the features in this field [10].

Advantages of AltBeacon

- This is an open source protocol;

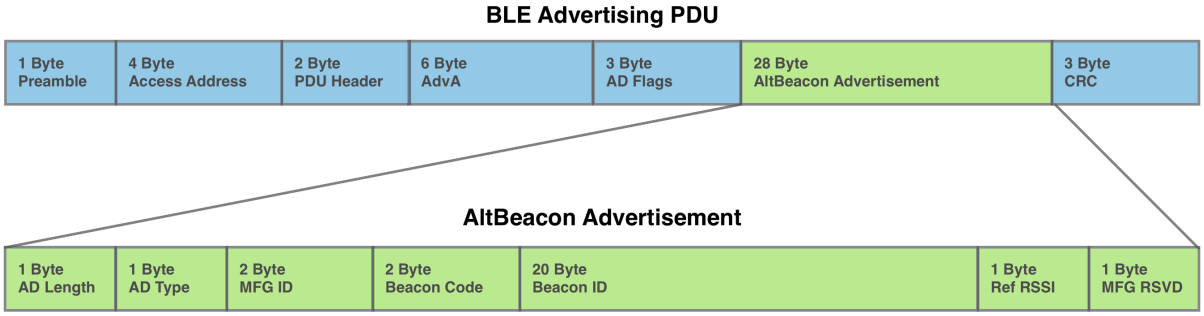


Figure 2.4: AltBeacon Packets [10]

- AltBeacon is compatible with other mobile operating platforms;
- AltBeacon is more flexible with a customisable source code.

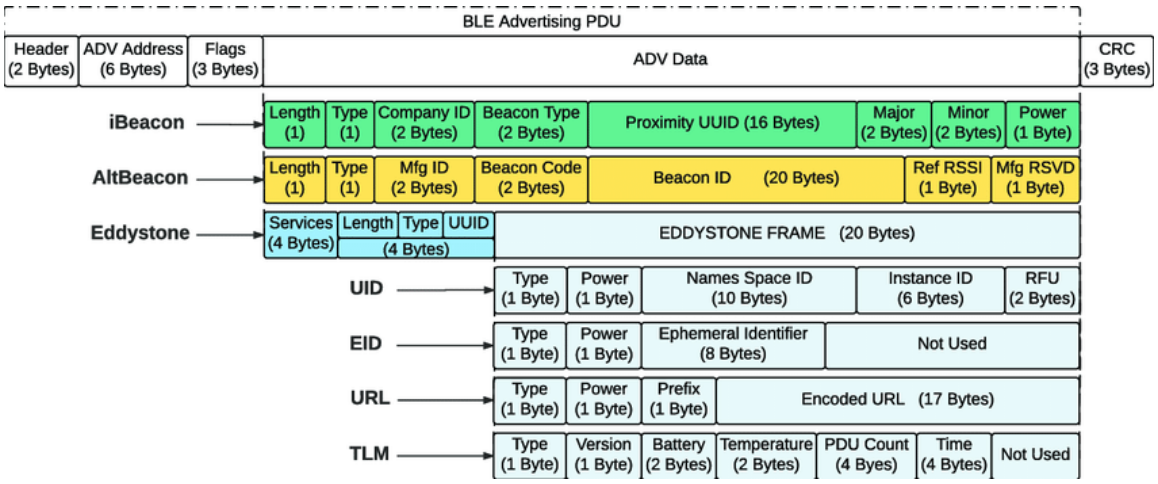


Figure 2.5: Structure of Beacon Protocols [11]

2.3 Beacon Market

In the Internet of things (IoT) market,there are various beacons available. Along with the variety of beacons, there are also various vendors who have a big role in making the beacon market competitive. They are competing for positioning their products in different applications in terms of lower prices, smaller sizes and longer battery lifes. Here, we have listed some of the notable vendors who have been in front foot in this process.[12]

- Estimote
- BKON
- RadBeacon Dot
- Konkat
- OpenBeacon BL
- BlueSense
- BlueCats
- RedBear Beacon B1
- SWIRL
- Glimworm
- Gimbal

Estimote Inc. is a technology start-up building a sensor-based analytics and engagement platform. Estimote has succeeded in capitalizing on having its development and management team based in Poland in order to manage costs, while projecting an image of the ultimate Silicon Valley startup. They have beautiful products that compete with major film starlets in their regular appearance in magazine, blogs, and journals

Managing beacons using their iOS app is very straightforward, with simple access to the controls to change all beacon parameters. They have added UUID rotation and support for Eddystone to compete with other tier one vendors. Given the huge volume of developers on their platform, they have a solid basis for relationships with the OS vendors.

BluVision are one of the contenders for the title of "beacon vendor that has sold the most product," with over one million beacons shipped in 2014.

Their hardware product portfolio is comprehensive with powered beacons and battery powered beacons with a sizable capacity. They also provide a Wi-Fi to Bluetooth bridge for remote management of beacons. BluVision has invested in its

software stack. They have experience with fleet management of large numbers of beacons; they have the conditional access layer that is required to control who can see a beacon.But their brand lacks the slick graphics and playful industrial design of Estimote [13].

Konkat.io was founded in 2013. They developed the first enterprise graded Bluetooth LE beacon supporting iBeacon and Eddystone. It has established its strength in the European market, where it has a leadership position. Kontakt.io's presence is becoming more established in the United States.

Kontakt.io was among the first vendors to support both Eddystone and iBeacon protocols. While their industrial design is not as conspicuous as Estimote, they have a well organized developer program with the most regular production of video content to guide entrants to the beacon system [1].

2.4 Comparitive table for commercial beacons

Above listed were the various manufacturers of the beacons. Now we have created a table 2.1 and its continuation on 2.2 that gives comparitive analysis of the various manufacturers of beacons along with their features, costs, protocols etc.

Beacon Manufactures	Built-in Radios	Image	Default battery life	Maximum Bluetooth range	Recommended SDK
Estimote[14]	Bluetooth 4.0 Bluetooth 5.0		1-3 years	7-200 meters	Estimote SDK Web IDE
Kontakt[15]	Bluetooth 4.2 compliant		2 year	up to 70 meters	Android SDK
RadBeacon Dot [16]	4.0 (Bluetooth Smart)		145 days	5m to 50m	FlyBuy SDK Proximity Kit SDK
OpenBeacon BL	BLE		not known	6 meters	Android SDK
BlueSense[17]	Bluetooth 4.0 LE		not known	more than 200 feet	free SDK
RedBear Beacon B1 [18]	Bluetooth 4.0 LE		4500+ hours	50m	not known
Gimbal	Bluetooth 4.0 LE		4 months	50m	Gimbal SDK v3/v4

Table 2.1: Comparitive table for commercial beacons.

Beacon Manufactures	Programming language	Thickness	Length	Width	Weight
Estimote	Objective-C Swift	6-27mm	29-99mm	28-99mm	8-98 gram
Kontakt	Java Swift	8-22mm	43-70mm	43-70mm	35-71 gram
RadBeacon Dot	Java Swift	15mm	35mm	35mm	14.7 gram
OpenBeacon BL	Kotlin Java	14-20mm (approx)	50-60mm	25-50mm (approx)	15 gram
BlueSense	java Swift Kotlin	15mm (approx)	35mm (approx)	35mm (approx)	20-60 gram
RedBear Beacon B1	Java	18.3mm	70mm	25mm	15 gram
Gimbal	Java	5.5mm (approx)	40mm (approx)	28mm (approx)	6.52 grams

Table 2.2: Comparitive table for commercial beacons(cont.)

2.5 Tools Used

We are using various tools for the development of our project. The tools that we have used for the development are :

2.5.1 Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It is built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.We choose Django because it is ridiculously fast and secure.

2.5.2 Java (programming language)

Java is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. We choose java programming language for the easiness of beacon configuration and also since it is one of the popular language it was easy for us to find the resources and ideas via internet.

2.5.3 REST API

An API is a set of definitions and protocols for building and integrating application software. A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding.

2.5.4 Retrofit

Retrofit is the a type-safe HTTP client for Android and Java. It makes our life way easier by providing us ability to communicate with the web server and get back the data from it in a nice formatted java objects.Retrofit is the class through which your API interfaces are turned into callable objects. By default, Retrofit will give you same defaults for your platform but it allows for customization.

2.5.5 AltBeacon Library

AltBeacon Library allows Android devices to use beacons much like iOS devices do. An app can request to get notifications when one or more beacons appear or disappear. An app can also request to get a ranging update from one or more beacons at a frequency of approximately 1Hz. It also allows Android devices to send beacon transmissions, even in the background.

Chapter 3

Approach / Modeling

3.1 Approach

Our project is mainly based on the application development. For the development of the application, Android and IOS are the two main leaders although most of the population of the world are Android users. We are developing our application for the Android users.

We are developing our application on the basis of native approach. Native apps uses native approach in which apps are built using platform-specific Software Development Kit (SDK)s and development tools provided by the platform vendors. This gives maximum access to the features and APIs available on each platform. We are using Java for Android. Before the development of mobile application, we have developed web page application built using Python framework namely Django for the backend management of our application.

3.2 Modeling

For the development of our application, we have created various models. Database Schema 3.3,Use Case Diagram 3.2, prototype of webpage and application3.11. Below we have explained about each prototypes used and their respective images.

3.2.1 Architectural Schema

The user has a phone with bluetooth and app installed in it. The beacon device is present in certain environment which is everytime advertising the UUID. The application when installed on the phone and bluetooth is turned on, the application is looking for the UUID and whenever it detects the UUID the application wakes up. The beacon data is sent to the internet.

The internet requests the data to the database with the beacon UUID. The backend webserver then reads the data and writes to the database. The database publishes it through the internet and sends in the form of notification to view in the application as shown in the figure 3.1.

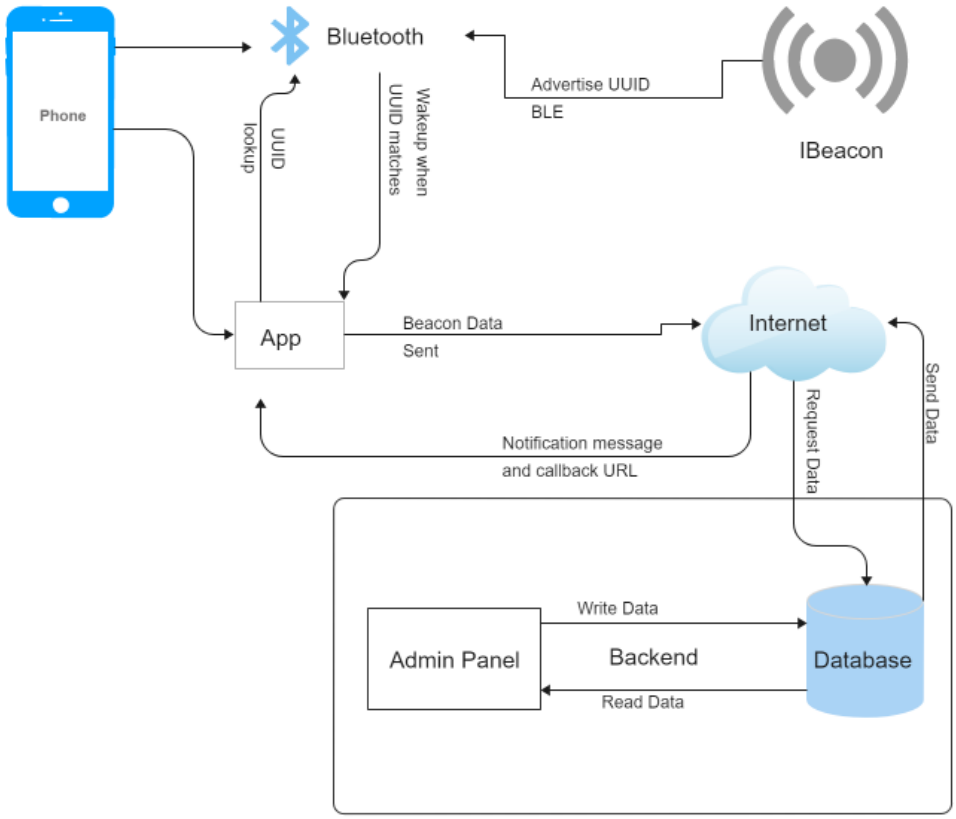


Figure 3.1: Architectural Schema

3.2.2 Use Case Diagram

Basically, Our application requires login and register by the guest then after being successfully logged in it demands user to enable Bluetooth and location on their android mobile phone. With this action the application will be able to detect beacon and only after the beacon is detected application is able to display the available notification on the screen. We have web page back-end developed using Django rest framework where we can manage beacons position, category, information, user and beacons. Admin plays a vital role to manage these activities.

According to the figure, Here guest is the actor who has the functionality of performing login and register in the Ipbeacon app. During the process of login the password is automatically verified by the system app and it displays login error if the password is not correct or if the user doesn't exist. After he is successfully registered and logged in he will have the functionality of user, User is the actor of the system who can perform functionality like reading notification and managing Bluetooth and location. Unless the user doesn't enable the location and Bluetooth, It is not possible to detect beacon. We also have another actor admin which generalized the user actor. Main functionalities of Admin are to manage users, beacon positions, categories, beacon locations and information's.

Figure 3.2 is the representation of our Use Case Diagram.

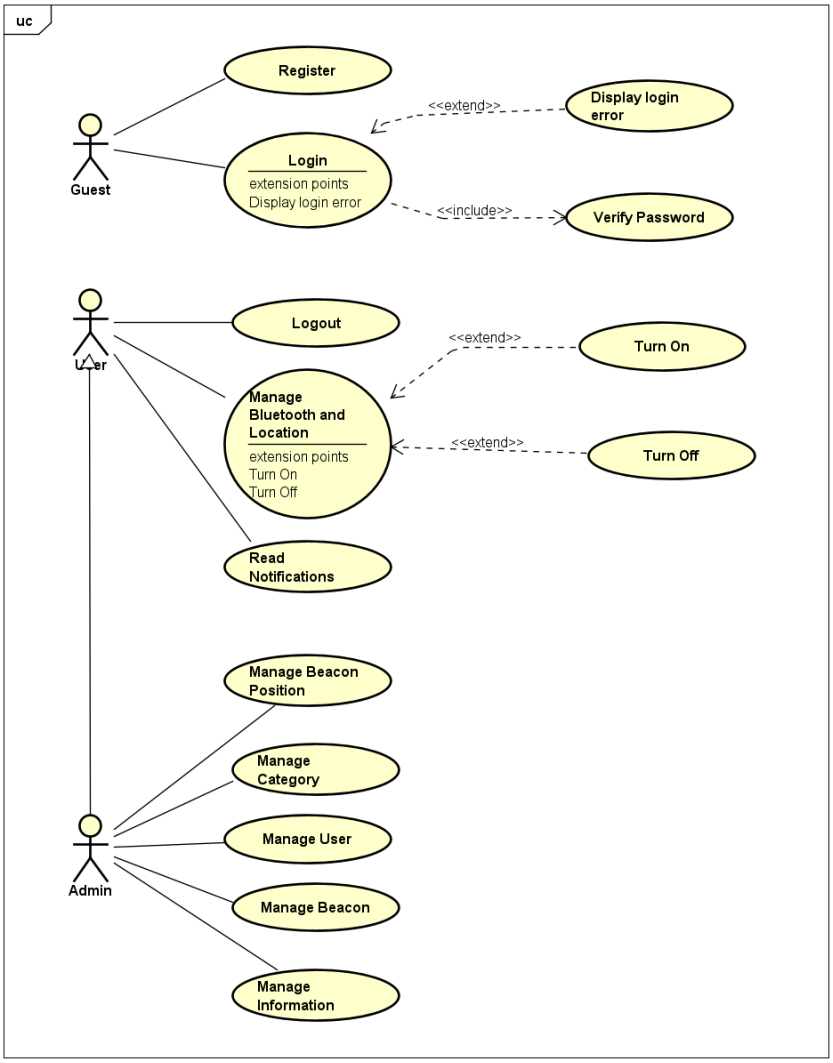


Figure 3.2: Use Case Diagram

3.2.3 Database schema

According to our schema, we have various tables like user, beacon, beaconposition, information and category.

We have user table with its attribute userID, userEmail, password, phoneNumber where userid is the primary key. We will also check if the user is admin who can manage everything. The next we have is beacon with its attribute beaconID, name and position. The beacon position stores the id of the original beacon from which it is contained to exactly locate the position of beacons. Since the beacon can be moved from one place to another according to the needs, we have added the table beacon position.

There is another table category to store the category where the beacon is positioned. For example: Canteen, Library, Classroom etc. We have a separate table for information that allows us to keep the information from the various positions where beacons are located. These are the information that are pushed in the form of notifications when beacon detects the user. Since, a beacon can have one or more information, datetime are used to distinguish them.

Figure 3.3 is the representation of our Database schema.

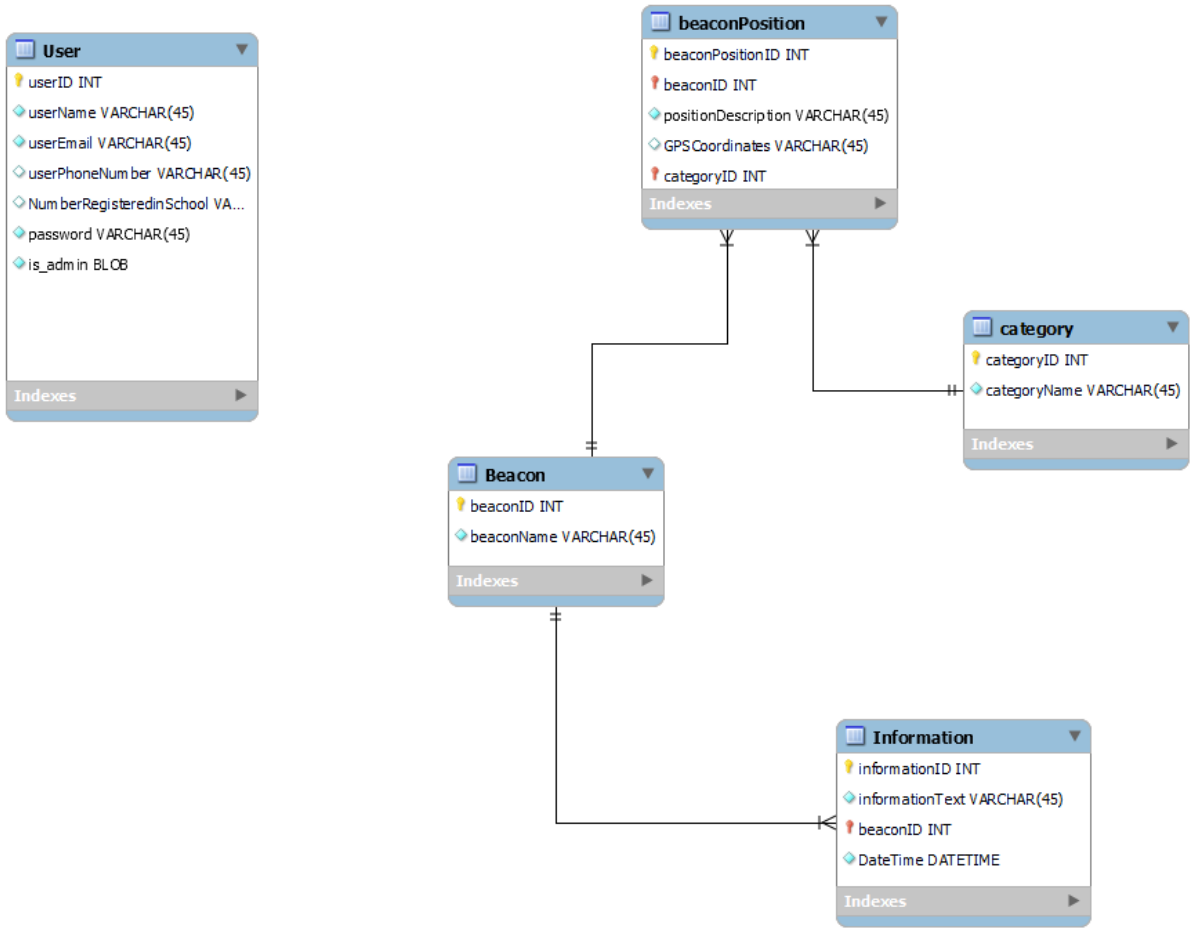


Figure 3.3: Database schema

3.2.4 Mockups of Webpage

Below here are the mockups for our web page. This web page is basically for manager or admin who manage the crud operation for all table maintenance of beacons and notification information.

User management page

At this page, the manager or admin can see the list of the users and also manages the Create,Read,Update,Delete (CRUD) operation of user. From the action column admin can view, edit and delete the particular user and also add users from the button add new

user.

Figure 3.4 is the representation of User Management Page.

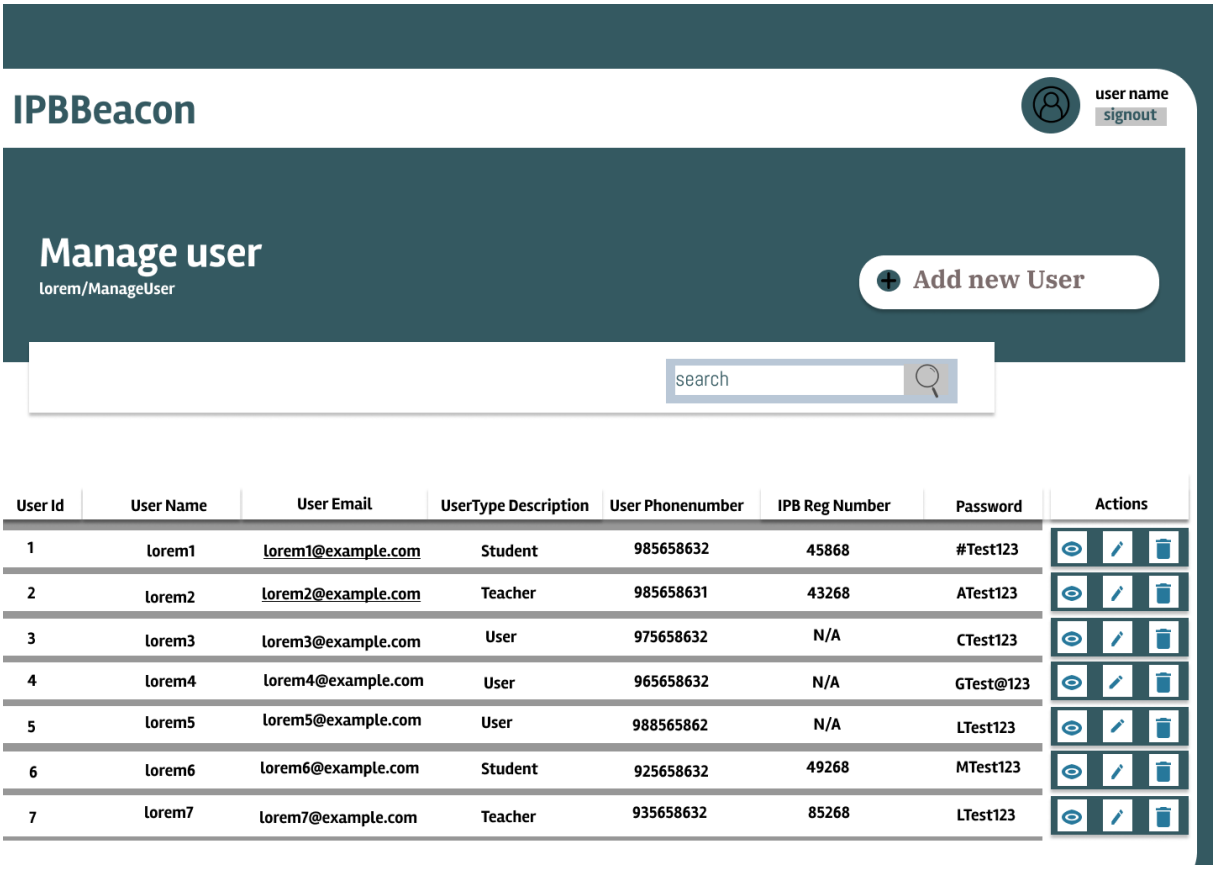


Figure 3.4: Mockup - User management page

Beacon management page

At this page, manager or admin can see the list of the Beacons and also manage the CRUD operation of Beacon.From the action column admin can view, edit and delete the particular beacon and also add beacons from the button add new beacon.

At figure 3.5, we display the Beacon management page.

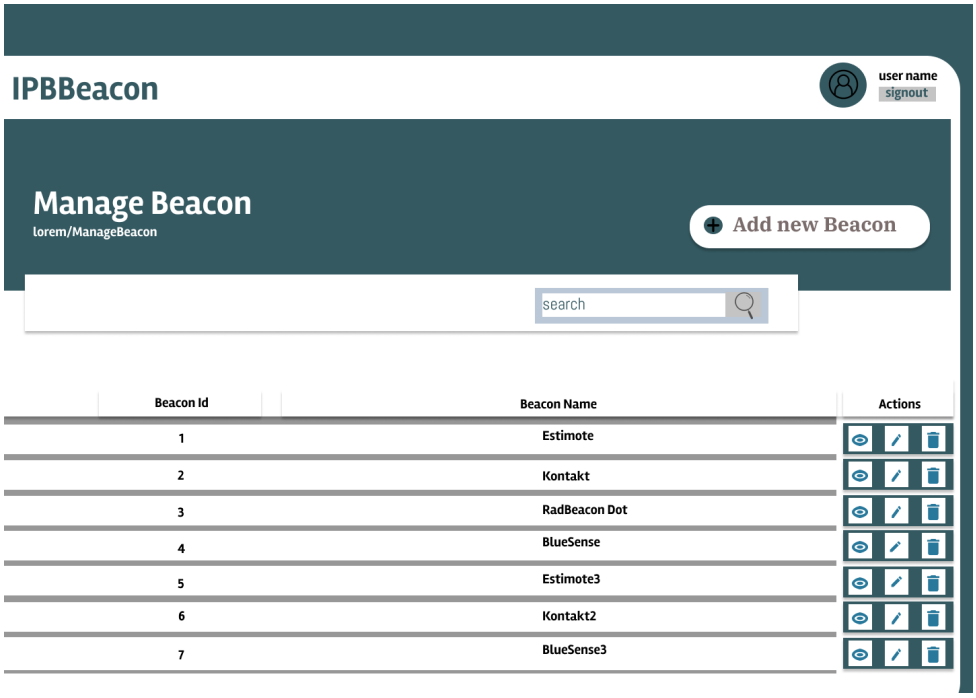


Figure 3.5: Mockup - Beacon management page

Beacon position management page

At this page, manager or admin can see the list of the position of Beacon and also manage the CRUD operation.From the action column admin can view, edit and delete the particular beacon position and also add beacon position from the button add new beacon position.

At figure 3.6, we display the Beacon Position Management page.

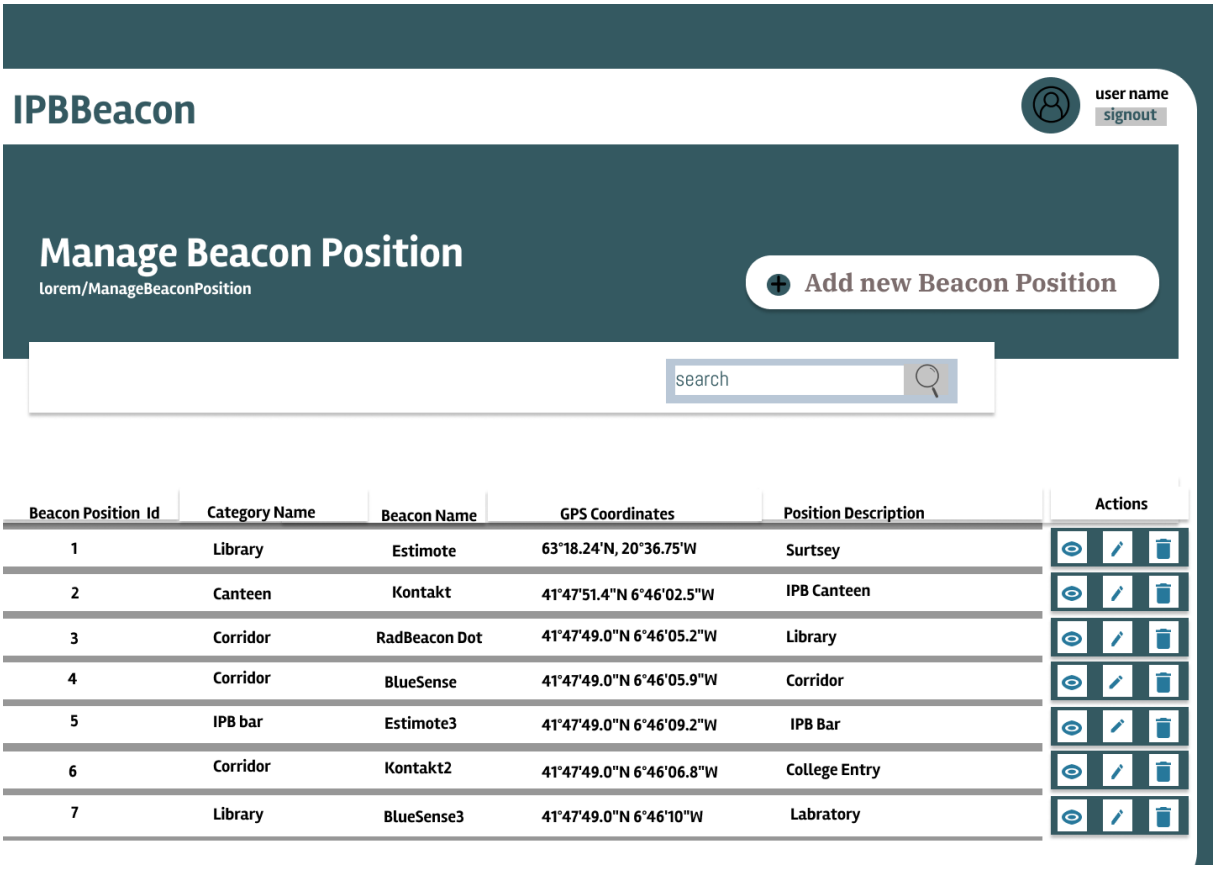


Figure 3.6: Mockup - Beacon position management page

Add New User Page

At this page, manager or admin can add new user. When they are adding new user, they should fill all the required field.

At figure 3.7, we display the Add New User Page.

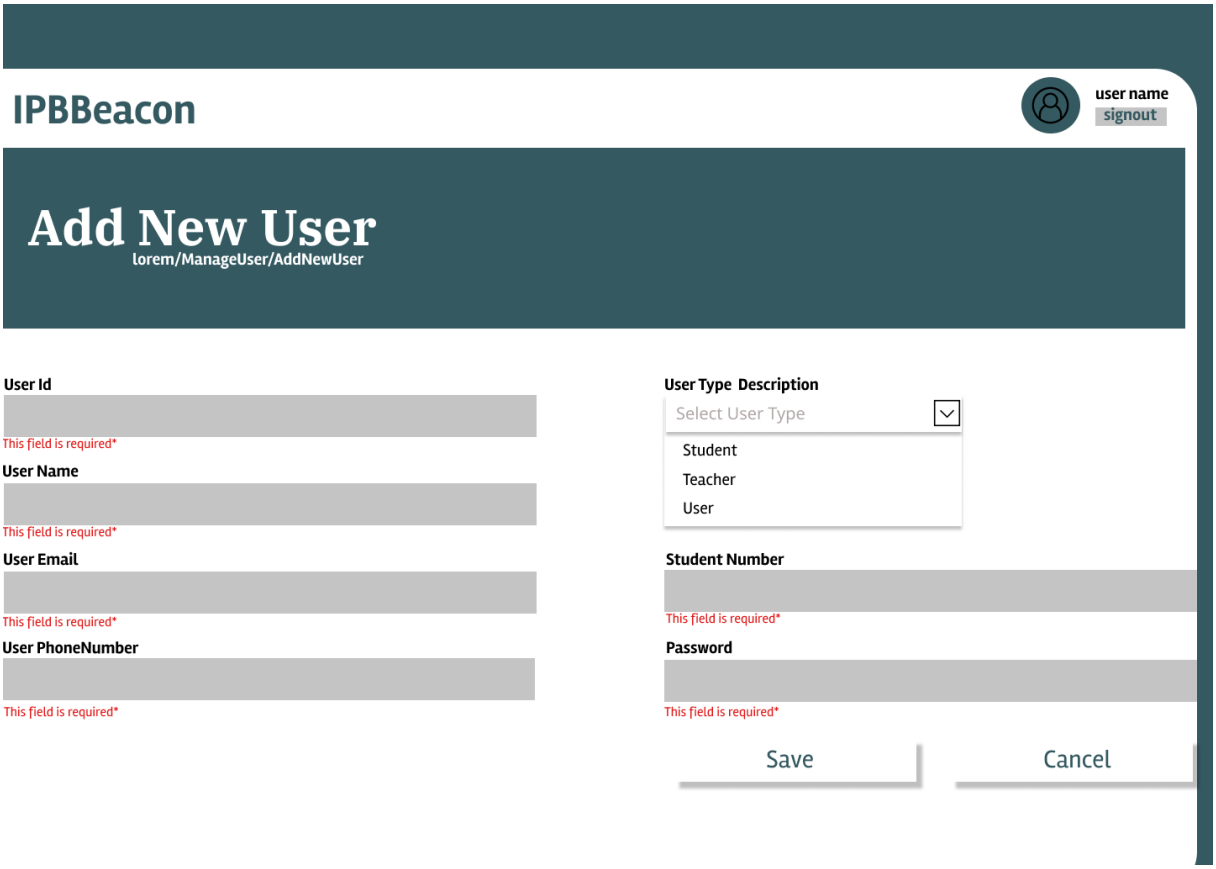


Figure 3.7: Mockup - Add New User Page

Add New Beacon Page

At this page, manager or admin can add new Beacon. When they are adding new beacon, they should go through all the required field.

At figure 3.8, we display the Add New Beacon Page.

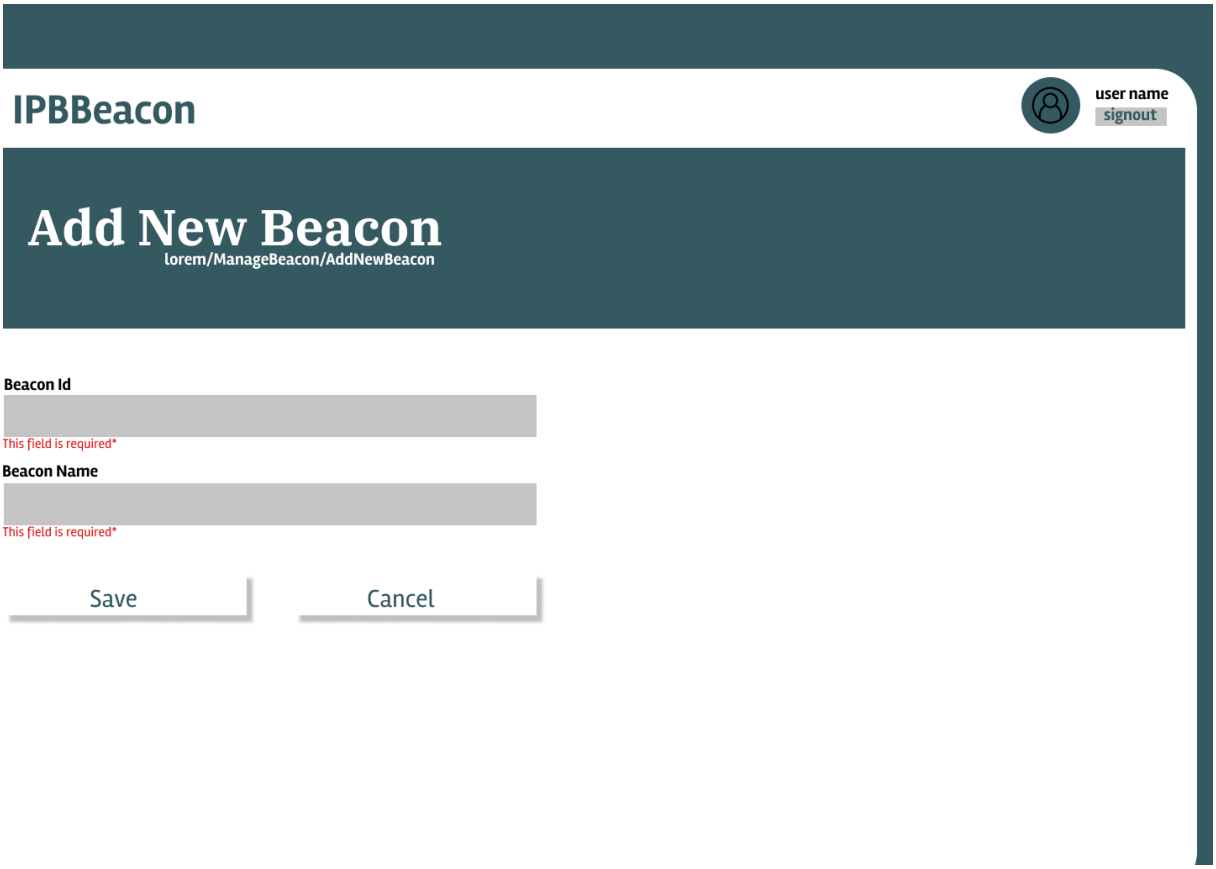


Figure 3.8: Mockup - Add New Beacon Page

Add New Beacon Position Page

At this page, manager or admin can add new beacon position. When they are adding new beacon position, they should go through all the required field.

At figure 3.9, we display the Add New Beacon Position Page.

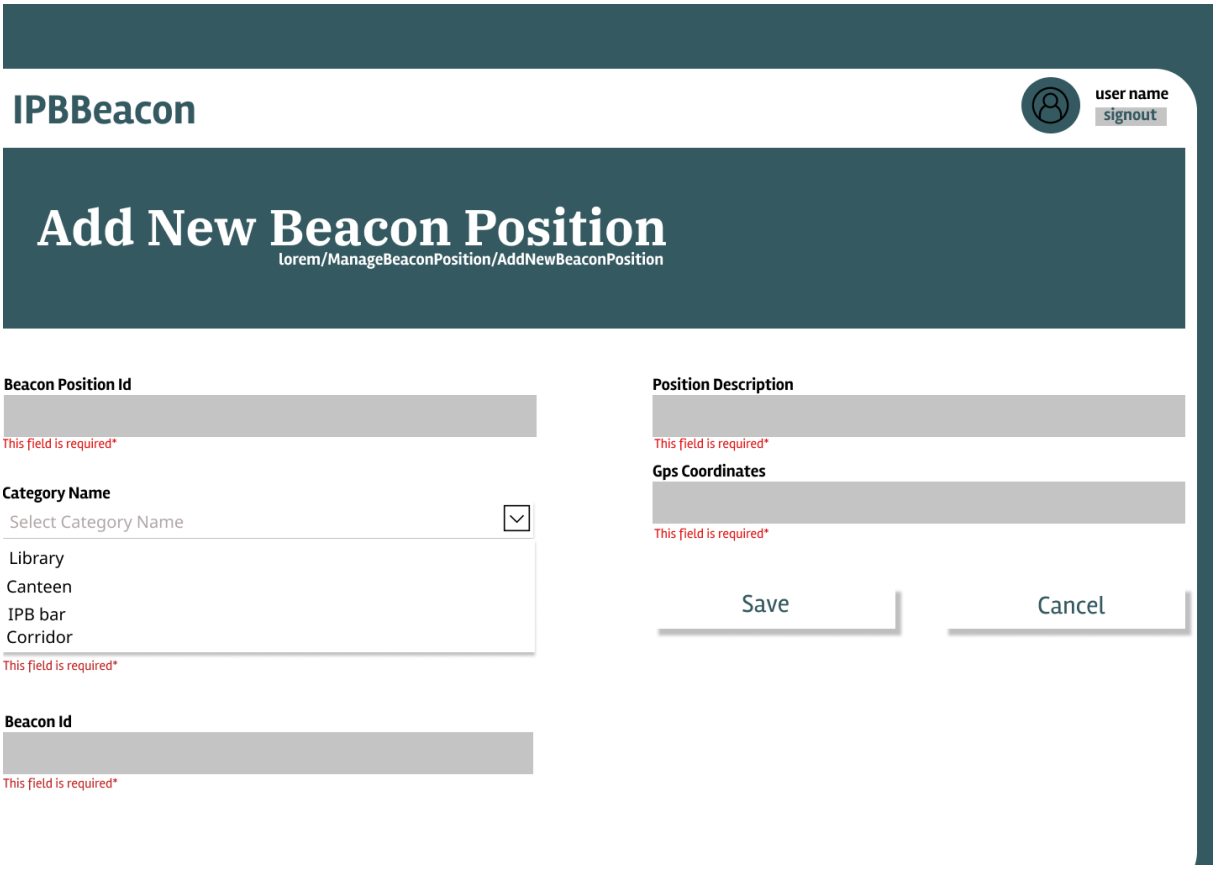


Figure 3.9: Mockup - Add New Beacon Position Page

3.2.5 Mockups of Application

In this section, we have created various prototypes of the mobile application to show how our application will look alike. We use these prototypes for the design of UI in our mobile application. Below we have represented every pages with their explanation of how our application works.

Landing Page

At this page, the users will be shown the information about that particular place as soon as any beacon is detected within that place, which has to meet the UUID of the beacon. At figure 3.10, we display the Landing Page.

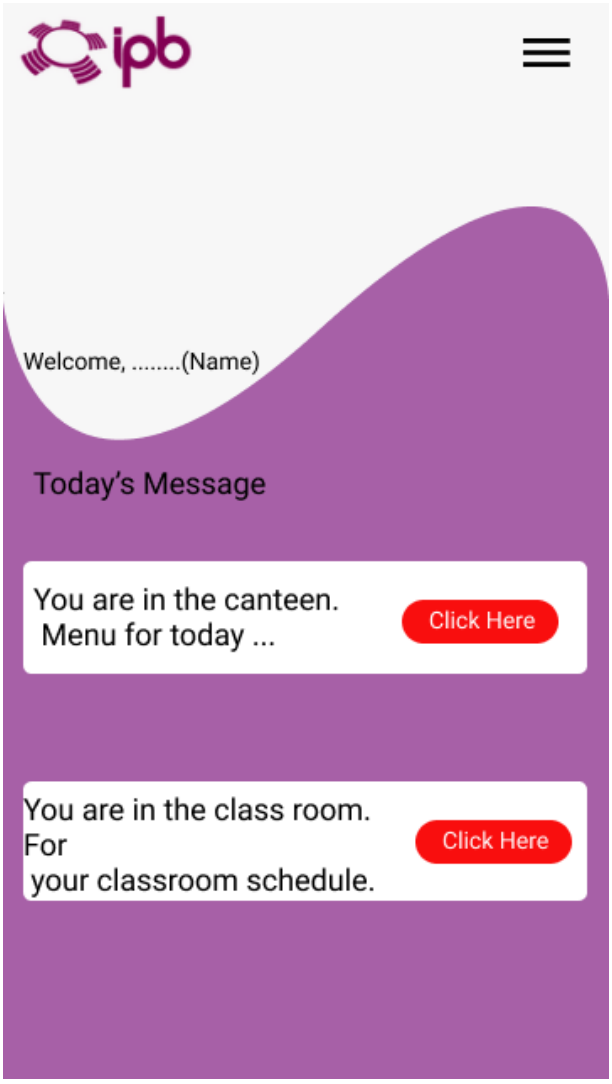


Figure 3.10: Mockup - Landing Page

Login Page

At this page, after the user opens the application (if not logged in), then the user gets into the login page. And the user will be able to Login if already registered. If he/she is not registered then he/she can register through the register page. To login he/she are asked the email and password with the help of which they registered. This page is for all the user who wants to visit this application.

At figure 3.11, we display the login page.

Figure 3.11: Mockup - Login Page

Register page

As our application is mainly focused on users who are from the schools using the registered number from school, we have a register page for users where the users will be able to input their registered number. Along with that number, they are asked to input their name, email, phone number and password for entry. Then, they are able to register and validate their account.

Figure 4.1, we display Register Page.

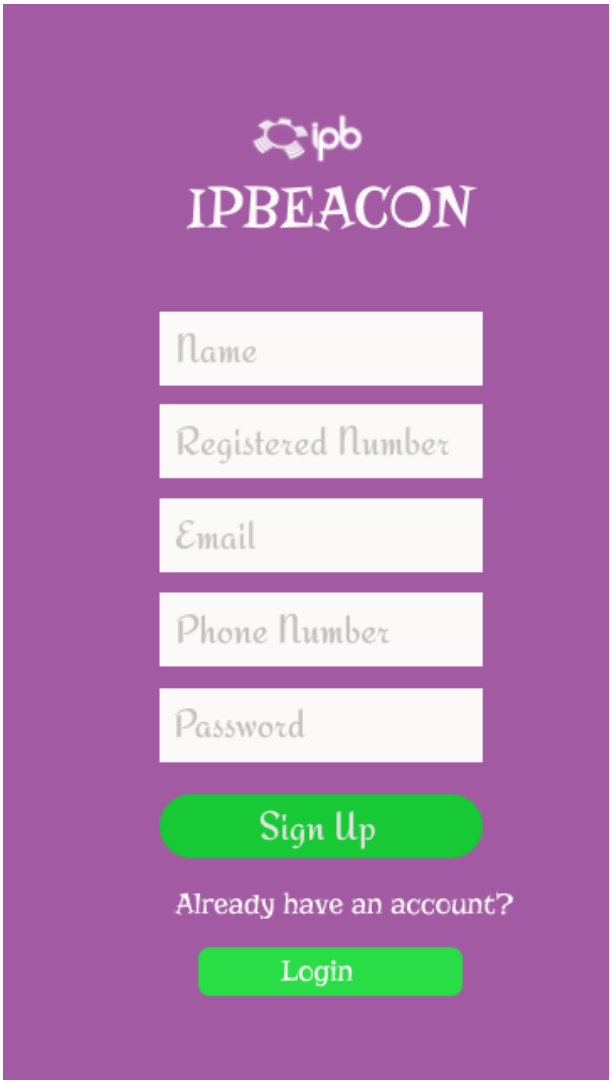


Figure 3.12: Mockup - Register Page

Chapter 4

Development

In this chapter, we will describe all the development aspects, the tools and libraries used for the development. We will also explain how our final product was developed.

4.1 Development

Our mobile application was developed in Java. We used Android Studio for the development of our application. Before the Android application development started, We had to create the backend to facilitate the Android application. Backend was created for the admin to manage the application.

Django was used for the development of backend webpage application. We used Django because it was very quick and easy to develop the backend unlike Yii which we had decided before. REST framework implementation in Django is also very easy and doable in quick way. This was the reason we chose Django rather than Yii.

At first, after Django app was created, We started by developing various models in models.py file according to the requirement. On the completion of creation of models, We then started the rest framework implementation. In the settings.py file, We first had to include the **rest_framework** for the application definition. Then, We created the serializers for each models and the respective view for these models excluding the user authentication. Different methods were followed for the user authentication. We used the

token based authentication system so that it will be easy for the server side to identify the user. The token based authentication is already included in the rest framework for the Django. To use it, we just added **rest_framework.authtoken** and performed the specific configuration in models.py file from the Django. This was the way how backend was developed.

```python
class Beacon(models.Model):
    beaconID = models.AutoField(primary_key=True)
    beaconName = models.CharField(max_length=10000,unique=True)

    def __str__(self) -> str:
        return str(self.beaconID)

class Category(models.Model):
    categoryID = models.AutoField(primary_key=True)
    categoryName = models.CharField(max_length=50)

    def __str__(self):
        return self.categoryName


class BeaconPosition(models.Model):
    beaconPositionID = models.AutoField(primary_key=True)
    beaconID = models.ForeignKey(Beacon, on_delete=models.CASCADE)
    positionDescription = models.TextField(max_length=45)
    GPSCoordinates = models.CharField(max_length=45)
    categoryID = models.ForeignKey(Category, on_delete=models.CASCADE)

    def __str__(self) -> str:
        return str(self.beaconID)
```

(a) models.py

```python
class BeaconPosition(models.Model):
    beaconPositionID = models.AutoField(primary_key=True)
    beaconID = models.ForeignKey(Beacon, on_delete=models.CASCADE)
    positionDescription = models.TextField(max_length=45)
    GPSCoordinates = models.CharField(max_length=45)
    categoryID = models.ForeignKey(Category, on_delete=models.CASCADE)

    def __str__(self) -> str:
        return str(self.beaconID)

class Information(models.Model):
    inforamtionID = models.AutoField(primary_key=True)
    informationText = models.TextField(max_length=10000)
    beaconID = models.ForeignKey(Beacon, on_delete=models.CASCADE)
    datetime = models.DateTimeField(max_length=45)

    def __str__(self):
        return self.informationText
```

(b) models.py

Figure 4.1: Django code series

```python
class BeaconSerializer(serializers.ModelSerializer):
    class Meta:
        model = Beacon
        fields = "__all__"

# category class serializer
class CategorySerializer(serializers.ModelSerializer):
    class Meta:
        model = Category
        fields = "__all__"

# beacon position class serializer
class BeaconPositionSerializer(serializers.ModelSerializer):
    class Meta:
        model = BeaconPosition
        fields = "__all__"

# information class serializer
class InformationSerializer(serializers.ModelSerializer):
    class Meta:
        model = Information
        fields = "__all__"
```

Figure 4.2: serializers.py

4.2 Application

Here , we have explained about how our application is developed and what are the objectives that are met.

For the first time, the users can login or register themselves if they arenot already registered to the application. The mobile application then acts to look for the beacon near the area. The application is scanning the beacons all the time even in the background. Whenever the beacon is detected, then the applications sends notification to the user and the relative information is shown to the user if the user wants. This was the main theme of our project.

4.2.1 Registration and Login

First of all, users will be required to provide login credentials which he/she used during registration. Once the details match, user will be logged in successfully and will be redirect to a page where we have display the username, email and IPB registered number of the logged-in user.

Our user registration activity contains 5 fields which are email, username, password, confirm password and registered school number. For registration user will be required to provide all data including username and email. Our user login activity contains 2 fields which are username and password. Users will be required to provide email and password to log in. When the login button is clicked user details will be validated against the existing store data. If it matches user will be redirected to the main activity which acts as our dashboard.

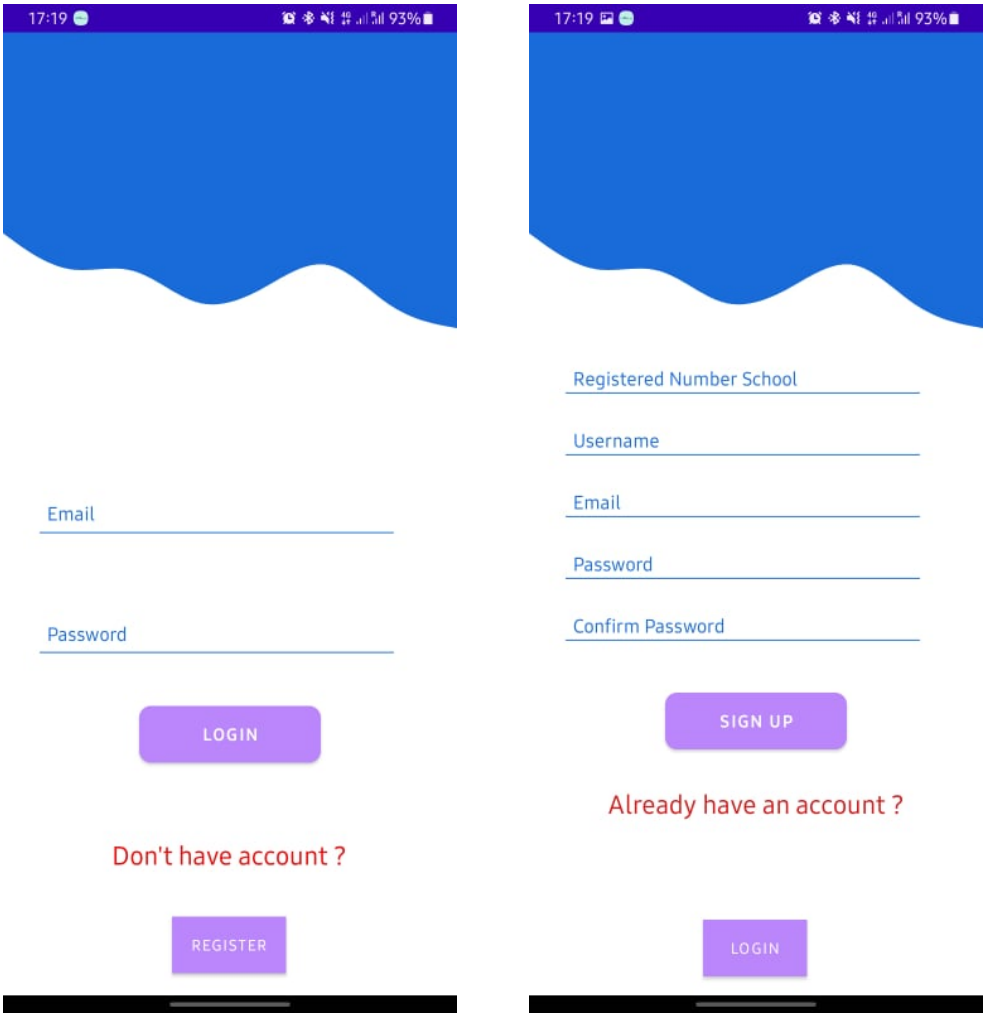
In order to read data from REST API, we add and configure retrofit, we begin by adding Gradle dependencies for retrofit, converter library and Http logger interceptor. The retrofit will let us perform network call and Http logger will log any outgoing and incoming request. Once retrofit is configured, we created an API client class that returns retrofit and user service where all the requests are added. Either post, get, delete, put. Below here is the screenshot of retrofit configuration4.3.

```
public class ApiClient {

    public static Retrofit getRetrofit(){

        HttpLoggingInterceptor httpLoggingInterceptor = new HttpLoggingInterceptor();
        httpLoggingInterceptor.setLevel(HttpLoggingInterceptor.Level.BODY);

        OkHttpClient okHttpClient = new OkHttpClient.Builder().addInterceptor(httpLoggingInterceptor).build();


        Retrofit retrofit = new Retrofit.Builder()
                .addConverterFactory(GsonConverterFactory.create())
                .baseUrl("https://ipbeacon.herokuapp.com/")
                .client(okHttpClient)
                .build();

        return  retrofit;
    }
    public static UserService getServices(){
        UserService userService = getRetrofit().create(UserService.class);

        return userService;
    }
```

Figure 4.3: Retrofit Configuration

We started by designing and implementing login register screen and we implemented **findallviewbyid** for both login and register.Then we add and configure retrofit. In order to do so, first of all, we added retrofit library, Gson Converter library. Then, we create a login register model, Perform API declaration and retrofit configuration. In the end, we

Save/Send User data to our API, Show API response whether success or not, User login using created credentials, Show login message whether successful or not and finally on successful login redirect user to the dashboard.



(a) Login.xml (b) Register.xml

Figure 4.4: LoginRegister.xml file

4.2.2 Beacon Detection

Our application is developed to detect the Ibeacon. To detect the beacons, we have used Android beacon library [19]. By default, the Android beacon library only detects the

altbeacon. We have provided the layout using **BeaconParser** which makes the app to detect only Ibeacon, as shown in the figure 4.5

```
public  void onCreate(){
    super.onCreate();
    BeaconManager beaconManager = org.altbeacon.beacon.BeaconManager.getInstanceForApplication(this);

    beaconManager.getBeaconParsers().add(new BeaconParser().setBeaconLayout("m:0-3=4c000215,i:4-19,i:20-21,i:22-23,p:24-24"));
    beaconManager.setDebug(true);
```

Figure 4.5: IBeacon layout

We created a class BeaconDetectionActivity which is of Application class. All the activities related to beacons are coded here including the scanning of the beacons, notification from the beacons.

Using the beacon manager class and monitor notifier, application scans the beacon in the foreground and background. The code implementation is presented in the figure 4.6.

```
beaconManager.enableForegroundServiceScanning(builder.build(), notificationId: 456);

beaconManager.setEnableScheduledScanJobs(false);
beaconManager.setBackgroundBetweenScanPeriod(0);
beaconManager.setBackgroundScanPeriod(1100);

Log.d(TAG, msg: "Background Monitoring");
beaconManager.addMonitorNotifier(this);

for (Region region: beaconManager.getMonitoredRegions()) {
    beaconManager.stopMonitoring(region);
}

beaconManager.startMonitoring(wildcardRegion);

}
```

Figure 4.6: Backround detection

Monitor notifier interface is implemented which allows application to monitor the beacons. This interface has three methods:

didEnterRegion(Region region), **didExitRegion(Region region)** and **didDetermineStateForRegion(int state, Region region)**

The code implementation for these methods are presented in the figure 4.7

```java
@Override
public void didEnterRegion(Region region) {
    Log.d(TAG,  msg: "did enter region.");
    insideRegion = true;
    // Send a notification to the user whenever a Beacon
    // matching a Region (defined above) are first seen.
    Log.d(TAG,  msg: "Sending notification.");
    sendNotification();

}


@Override
public void didExitRegion(Region region) {
    insideRegion = false;


}

@Override
public void didDetermineStateForRegion(int state, Region region) {


}
```

Figure 4.7: Monitor Notifier Methods

didEnterRegion method is called when at least one beacon is visible inside a region. Whenever the beacon is visible inside a region, using the **sendNotification** function, the notification is sent informing that beacons are visible.

4.2.3 Notification

After the beacon is detected by the application, a notification is pushed solved in method **sendNotification** using the Notification.Builder class which is the builder class for notification in Android.

At first, notification channel is created using **NotificationChannel** class as we have to assign each notification to a unique channel id. All the behaviour of our notification is also customized in the notification channel class. Then, using the notification builder this channelid is assigned and notification is built like in the figure shown 4.8. When tapped to the notification, the user is redirected to the **HomeActivity** page where the users can see the various information related to them.

To customize more in the application, we have used the notification in such a way that it is always shown which gives the user the information that the application is scanning for the beacons everytime as shown in the figure 4.9.

```
private void sendNotification() {
    NotificationManager notificationManager =
            (NotificationManager) this.getSystemService(Context.NOTIFICATION_SERVICE);
    Notification.Builder builder;
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        NotificationChannel channel = new NotificationChannel( id: "IPBeacon Notifications",
                name: "IPBeacon Notifications", NotificationManager.IMPORTANCE_HIGH);
        channel.enableLights(true);
        channel.enableVibration(true);
        channel.setLockscreenVisibility(Notification.VISIBILITY_PUBLIC);
        notificationManager.createNotificationChannel(channel);
        builder = new Notification.Builder( context: this, channel.getId());
    }
    else {
        builder = new Notification.Builder( context: this);
        builder.setPriority(Notification.PRIORITY_HIGH);
    }
    TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
    stackBuilder.addNextIntent(new Intent( packageContext: this, HomeActivity.class));
    PendingIntent resultPendingIntent =
            stackBuilder.getPendingIntent(
                    requestCode: 0,
                    PendingIntent.FLAG_UPDATE_CURRENT
            );
    builder.setSmallIcon(R.mipmap.ic_launcher_round);
    builder.setContentTitle("Beacon Detected");
    builder.setContentText("You have important notification incoming. Tap here to view");
    builder.setContentIntent(resultPendingIntent);
    notificationManager.notify( id: 1, builder.build());
```

Figure 4.8: Notification Defined

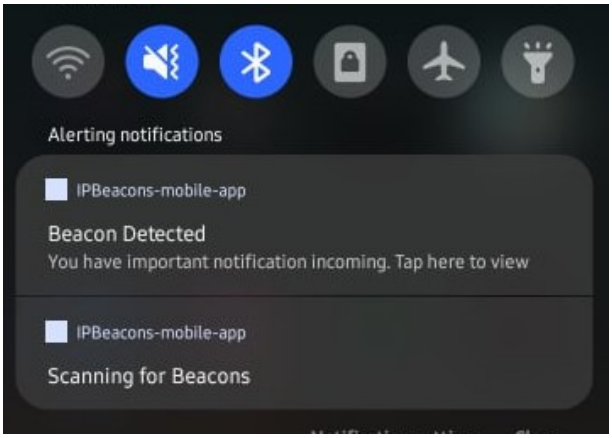


Figure 4.9: Notification

4.2.4 Getting Information

The main thing in our application is to get information from the cloud after the beacon is successfully detected by the application. We have implemented retrofit to get the data from the web server rather than directly implementing database in the application. The activity has been implemented in **HomeActivity.java**. The implementation can be seen in D.

Even before the beacon is detected, we have the functions in **HomeActivity.java** which checks whether the bluetooth is activated or not since the bluetooth is required for the application to detect the beacon. Also, the location permission is check as well. The first time the application is installed on the mobile phone, it gives alert informing to enable the location permission. Only doing these makes the application to detect the beacons and get the information. These functions are implemented with the name **verifyBluetooth()** and **requestPermissions()**. These two functions were created manually.

After the bluetooth and location permission result, we have a function **onResume()**. This function is called when the application is in forground state. Inside this function, we have implemented the main part of our project that is to get the information from the cloud through the filtering from the webserver. Inside this function, we have **didRangeBeaconsinRegion** function which includes the collection of beacons as parameters. This function is implemented when beacon becomes visible and send the signal.

beacons.size() gives the number of beacons detected by the application. Only if any of the beacons are detected, we go through the collection of beacons in the loop and send the request to the server to get the information of the particular detected beacon . The beacon id that the application detects, we can get it from **b.getId(1)** as referenced in figure 4.10

To get the useful information from our webserver, we have first created a interface

```
for(Beacon b:beacons){
    Call<List<Info>> infoCall = ApiClient.getApi().getInfos(b.getId1().toString());
```

Figure 4.10: detected beaconid

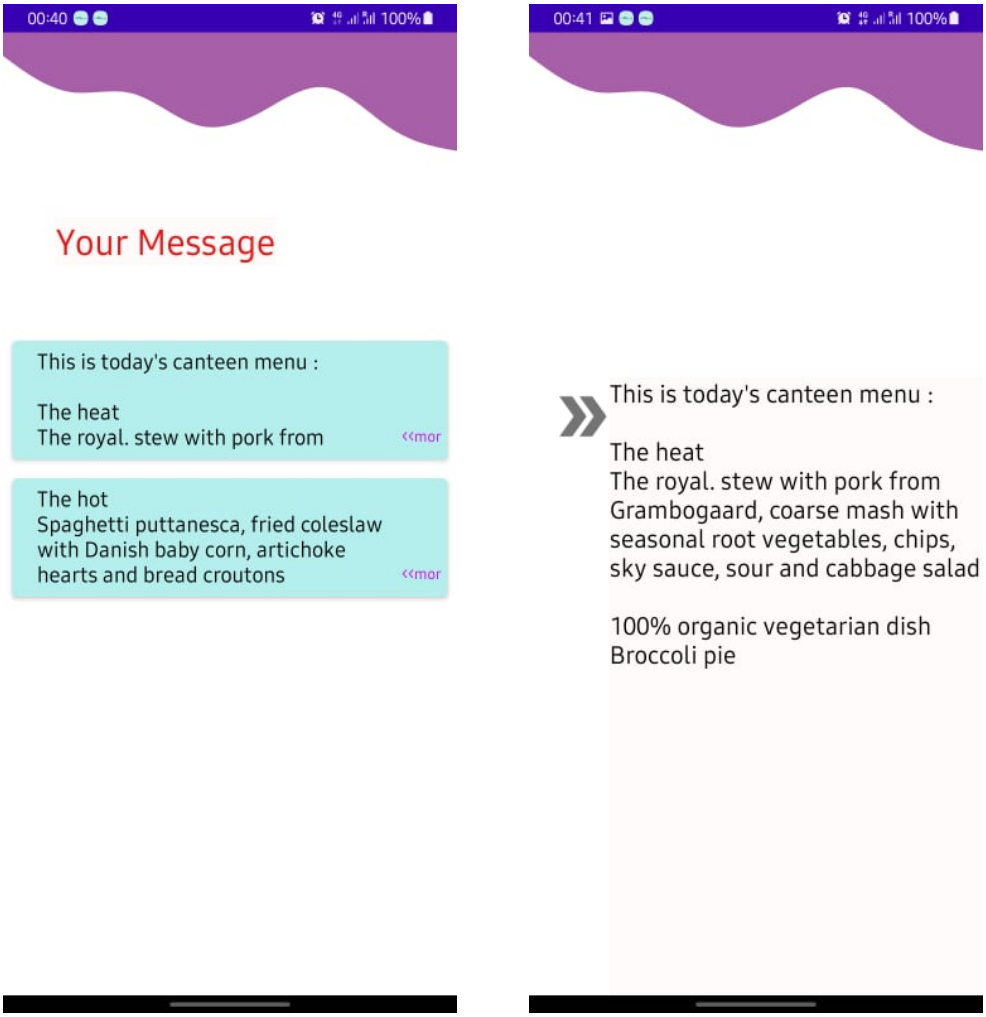
namely **JsonPlaceHolderApi.java** which is the interface for get request to the server. We also have **ApiClient.java** class where the implementation of retrofit is done. The REST endpoint is also implemented in the same class. Then, the model is created where the getters are implemented to get the information from the webservices. The API call is set in the **HomeActivity.java** itself referenced in figure

```
Call<List<Info>> infoCall = ApiClient.getApi().getInfos(b.getId1().toString());

infoCall.enqueue(new Callback<List<Info>>() {
    @SuppressLint("SetTextI18n")
    @Override
    public void onResponse(Call<List<Info>> call, Response<List<Info>> response) {
        if(response.isSuccessful() && response.body()!=null){

            infoList.clear();
            infoList.addAll(response.body());
            adapter.notifyDataSetChanged();
            progressBar.setVisibility(View.GONE);
        }
```

Figure 4.11: API call

When we get the information after application detects the beacons, they are listed in a recycler view , image referenced 4.12a

The information can be viewed seperately on the basis of their id when they are listed in the homepage of the application as shown in the figure 4.12b



(a) Information Prototype (b) Information onClick

Figure 4.12: Information

Chapter 5

Tests/Discussion

In this chapter, we will discuss the tests and result of our application.

5.1 Tests

The application was made to run for the Android devices. We tested the application to be sure if it runs on the emulator and physical devices. When we run the application on the emulator, We got the problem in detecting the beacons from the application through emulator. Initially, We tried to work with the emulator only but then after we got no positive results, We tried on our physical devices. On the physical devices we were successful in achieving the results. We later on went on to find the answer to our problem regarding the emulator that the additional part of coding has to be done to be able to run the application in the emulator. **But Our application only runs on physical devices.**

The initial objectives that is to get the information from the cloud on the beacon detection was successfully fulfilled. But to test more precisely, We first tested with only one beacon and we got the information properly. Then we tested using two beacons at the same time, but then the result that we got was different. The information that we get from both beacons were shown at once but not together. Our application is developed

in such a way that the beacons are located in the different positions. So, when at the same time both beacons are detected, then the information gets overlap and we see the information from one of these two beacons.

As we were also provided the beacons by our supervisors, we also tested our applications using those devices. The image can be seen at B.2

Our application solves the problems discussed in the chapter 3 and the final objective of the project was fulfilled properly.

5.2 Difficulties / Discussion

At the intital phase of the project development, We had decided to develop the mobile application using Kotlin. Since we had no experience working with the beacons and also the Android application using Kotlin, it was very difficult to figure out how it's supposed to be done. We then created the application using Kotlin at first but looking after the proper resources and Google searching, We found out that Java is a better option, although Kotlin was designed to interoperate fully with Java. Very less resources are available for the beacon application. Since, We were new to mobile application development and we had some knowledge about JAVA programming language, We then started to work with java.

We faced difficulties while developing the mobile application. We had developed the webpage using Yii because we both already had knowledge about the framework. But while implementing the REST API it was difficult for us because we never tried it before. We tried to implement the REST using Yii but we got to know that Yii is not the proper solution to our problem. Then, We switched to Django which was pretty fast and reliable method to implement the backend and the REST API implementation was also very easy. We tested our solution and it worked pretty comfortably. So, in contrast to what was decided in the first part of the project to develop with Yii, We switched to Django for

the proper and fast implementation. During this course, We were able to differentiate between Yii and Django framework.

Then in the mobile application, at first we faced problem with detection of beacons. We were unsure how it's done. Then we got to know that there is a library named Android beacon library which designed for the same solution [19]. We went on to do the more research on this and started implementing. The solution worked then. We were successful in achieving what we wanted the application to do. We tested using the beacon devices provided by our supervisor and we were successful in detecting the beacons. The reference is attached in figure 5.1

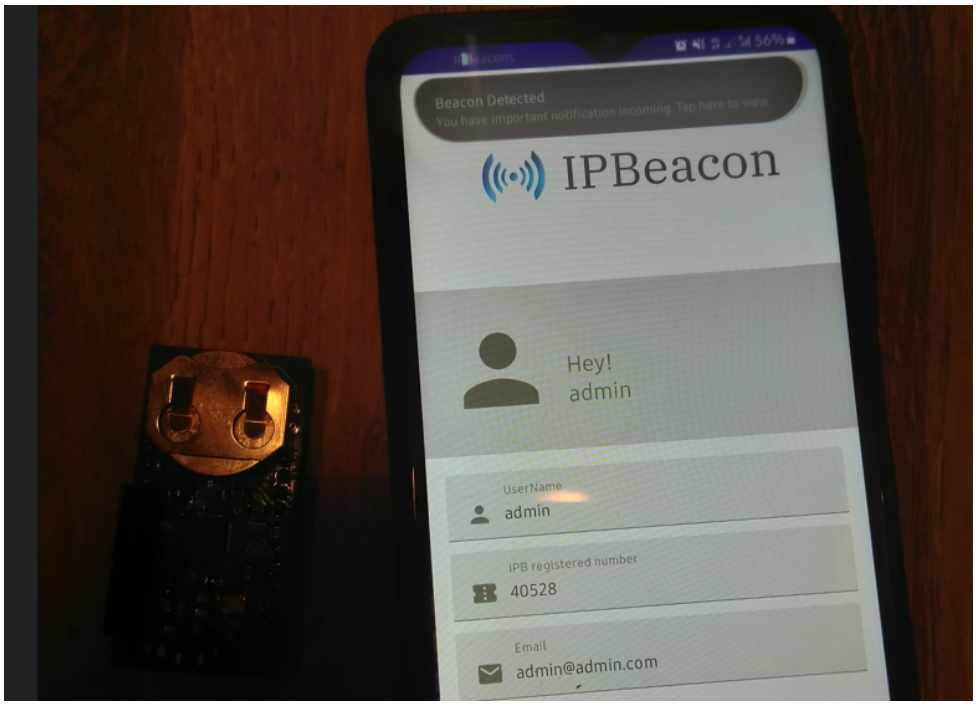


Figure 5.1: Successful beacon detection

Chapter 6

Conclusions

Due to the rapid development in the field of technology, the use of smartphones has been the integral part of our lives. Many devices are being developed which makes the human life more comfortable. Due to the increased automation of daily tasks, the need to automatically notify the user with the interesting information was the useful discussion and with the help of BLE devices like beacon this is possible. So, Our project was developed with the aim of same objective.

Initially we were suppose to develop the application for both android and IOS but as the time passed by, We found difficult to provide the solution for both android and IOS as we got stuck in finding the the solution to develop in this new environment. Hence, We focused only on Android as the number of users for the Android devices are more than IOS and IOS required more resources for it to develop.

During the course of this project, We got the opportunity to learn many new things including the development of the full fledged mobile application which will help us in the future. To conclude, this project was a very nice project. We could gather the knowledge about IoT devices. This project also helped us to understand various different frameworks and technologies. Though through various difficulties, We were successful in completing the project as required.

6.1 Future Work

As the objective of the project has been met, there are things that can be done to improve the project further more. Firstly, the application can also be developed in IOS devices. The web server of our application can be customized further more with more advanced features like the live position of the beacons. We would also have liked to use our application in the real context.

Bibliography

[1] N. Newman, "Apple iBeacon technology briefing," *Journal of Direct, Data and Digital Marketing Practice*, vol. 15, no. 3, pp. 222–225, 2014, ISSN: 17460166. DOI: 10.1057/dddmp.2014.7.

[2] *Types of beacons - Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Types{_}of{_}beacons{\#}AltBeacon{_}(Radius{_}Networks) (visited on 11/13/2020).

[3] *What is iBeacon™? · Accent Systems*, en-US. [Online]. Available: https://accent-systems.com/support/faq/what-is-ibeacon/ (visited on 01/14/2021).

[4] *What is iBeacon? (Everything You Need To Know)*, Jun. 2017. [Online]. Available: https://buildfire.com/what-is-ibeacon/ (visited on 01/14/2021).

[5] D. Torstensson, "Indoor Positioning System Using Bluetooth Beacon Technoology," *MIS Quarterly Vol.*, vol. 34, no. 2, pp. 261–279, 2010.

[6] *BLE Beacons: iBeacon, AltBeacon, URIBeacon and derivatives*. [Online]. Available: http://austinblackstoneengineering.com/ble-beacons-ibeacon-altbeacon-uribeacon-and-derivatives/ (visited on 11/25/2020).

[7] Matouš Havlena, "iBeacons - How do they (technically) work? - Matouš Havlena," 2017. [Online]. Available: http://www.havlena.net/en/location-technologies/ibeacons-how-do-they-technically-work/ (visited on 11/10/2020).

[8] *Eddystone - IoT Projects with Bluetooth Low Energy [Book]*. [Online]. Available: https://www.oreilly.com/library/view/iot-projects-with/9781788399449/19495b89-8a0a-43f8-8dfd-955bdcc203db.xhtml (visited on 11/25/2020).

[9] *GitHub - AltBeacon/android-beacon-library: Allows Android apps to interact with BLE beacons*. [Online]. Available: https://github.com/AltBeacon/android-beacon-library (visited on 11/25/2020).

[10] *BLE Beacons: iBeacon, AltBeacon, URIBeacon and derivatives – Austin Blackstone Engineering*. [Online]. Available: https://austinblackstoneengineering.com/ble-beacons-ibeacon-altbeacon-uribeacon-and-derivatives/ (visited on 01/14/2021).

[11] *Structure of iBeacon, AltBeacon and Eddystone beacon protocols. | Download Scientific Diagram*. [Online]. Available: https://www.researchgate.net/figure/Structure-of-iBeacon-AltBeacon-and-Eddystone-beacon-protocols{_}fig2{_}322092785 (visited on 11/25/2020).

[12] J. C. Ferreira, R. Resende, and S. Martinho, "Beacons and BIM models for indoor guidance and location," *Sensors (Switzerland)*, vol. 18, no. 12, 2018, ISSN: 14248220. DOI: 10.3390/s18124374.

[13] *IoT Applications - Internet of Things & NFC Technology - HID Global*. [Online]. Available: https://www.hidglobal.com/internet-of-things?utm{_}source=bluvision.com{\&}utm{_}campaign=bluvision.com{\&}utm{_}medium=301 (visited on 11/25/2020).

[14] *iBeacon Technology and Estimote's Bluetooth Beacons | Ibeacon technology, Beacon technology, Ibeacon*. [Online]. Available: https://ar.pinterest.com/pin/365847169717057007/ (visited on 11/24/2020).

[15] *Kontakt.io Store*. [Online]. Available: https://store.kontakt.io/ (visited on 11/24/2020).

[16] *Amazon.com: RadBeacon E4 - All-Weather, Long-Life Bluetooth Smart Proximity Multi-Beacon Using iBeacon and AltBeacon Technology: Computers & Accessories*. [Online]. Available: https://www.amazon.com/RadBeacon-All-weather-Long-life-Multi-beacon-Technology/dp/B00R9QC4A2 (visited on 11/24/2020).

[17] *blueSense - Crunchbase Company Profile & Funding*. [Online]. Available: https://www.crunchbase.com/organization/blue-sense-networks (visited on 11/24/2020).

[18] *Red Bear BLE iBeacon: Amazon.ca: Electronics*. [Online]. Available: https://www.amazon.ca/CE0197-Red-Bear-BLE-iBeacon/dp/B01C95SFDC (visited on 11/24/2020).

[19] *Android Beacon Library*. [Online]. Available: https://altbeacon.github.io/android-beacon-library/ (visited on 09/08/2021).

Appendix A

Original project proposal

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Curso de Licenciatura em Engenharia Informática
Projeto 3º Ano - Ano letivo de 2020/2021

IPBeacons at Campus – Mobile Application

Supervisor: Pedro Filipe Fernandes Oliveira

Co-supervisor: Paulo Matos

1 Objectives

It is intended the development and implementation of a solution that allows the sending of information through personalized notifications to the user, when he approaches the different campus locations, namely canteen, office, classrooms, bar, etc., in an non-invasive way and without any user interaction, using beacon devices.

2 Details

The human daily mobility, attached with the attractiveness of the increased automation of daily tasks, also comes in the context of daily mobility existing on a university campus, creating the need to automatically notify the user with interesting information.
In this context and to meet the automation of the whole process, it is intended to develop a solution that allows automatically and without any user interaction, the user detection when he is near a certain environment/space. To do this a beacon device must be present in the environment/space, and the user must have the application to develop on his smartphone.
Taking advantage of Beacon technology, an application should be developed to notify interest information to the user, depending on the user is. For instance, the canteen daily menu, classroom schedule, a bar promotion, and so on.

3 Project development strategy

1. Previous study of existing platforms;
2. Requirements analysis;
3. Drawing of smartphone application protoype;
4. Selection of programming language and possible frameworks to use;
5. Smartphone application development;
6. Implementation and testing, in real context;
7. Possible improvements after testing;

Team size: 1 to 2 students
Resources: PC, smartphone

Appendix B

Images

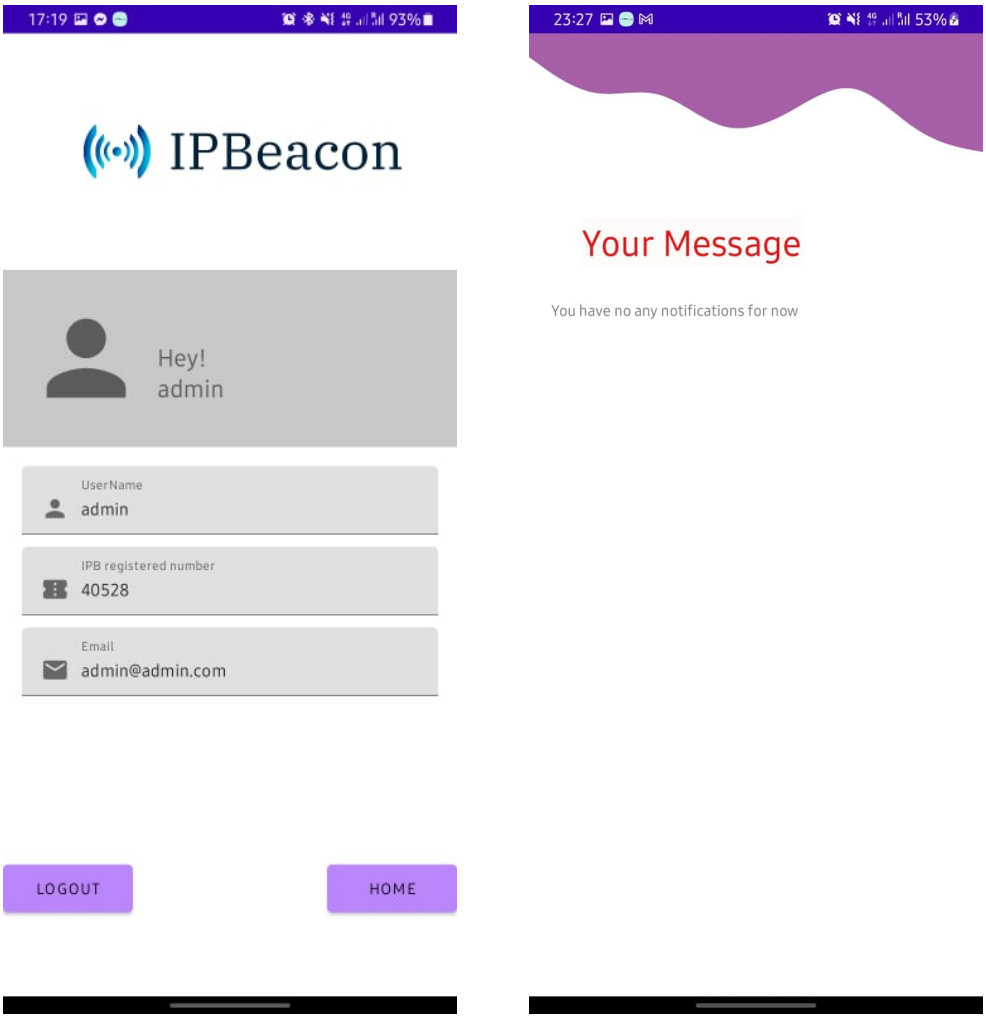
Listing B.1: iBeacon Packets

```
02 01 06 1A FF 4C 00 02 15 # iBeacon prefix (fixed)
B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D # Proximity UUID
00 49 # Major
00 0A # Minor
C5: 2's complement of measured TX power
```

Listing B.2: iBeacon Prefix

```
02 # Number of bytes that follow in first AD structure
01 # Flags AD type
06 # Flags value
1A # Number of bytes that follow in second (and last) AD structure
FF # Manufacturer specific data AD type
4C 00 # Company identifier code (0x004C == Apple)
02 # Byte 0 of iBeacon advertisement indicator
15 # Byte 1 of iBeacon advertisement indicator
```

B.0.1 Images from the Application Home Page and Notification Page



(a) Landing Activity (b) No Notification

Figure B.1: Application

B.0.2 Beacon device Image

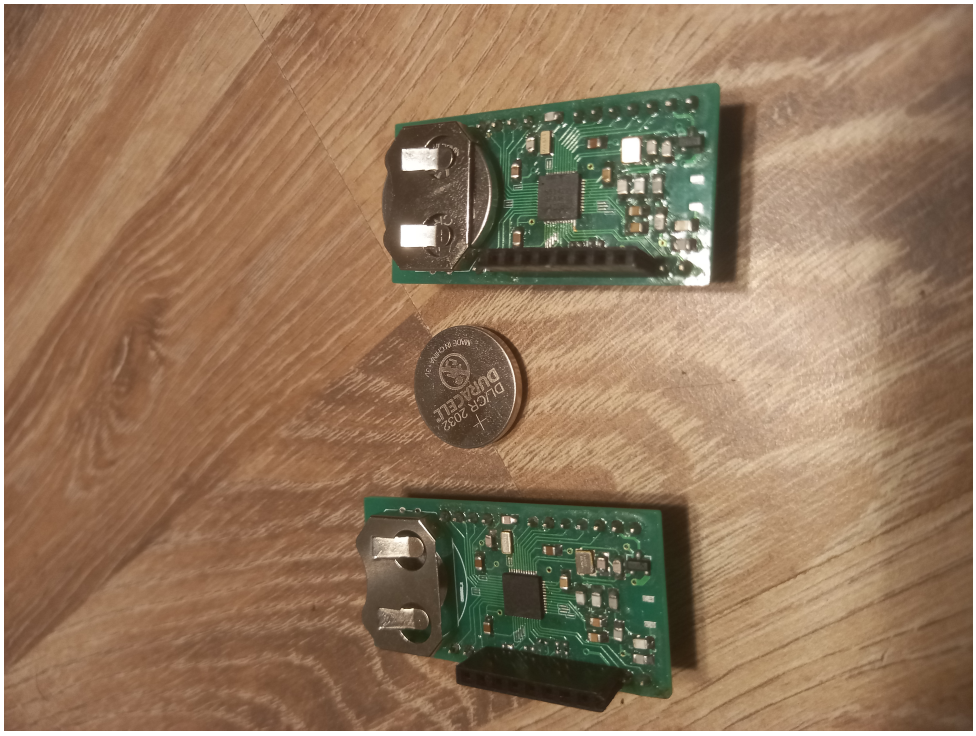
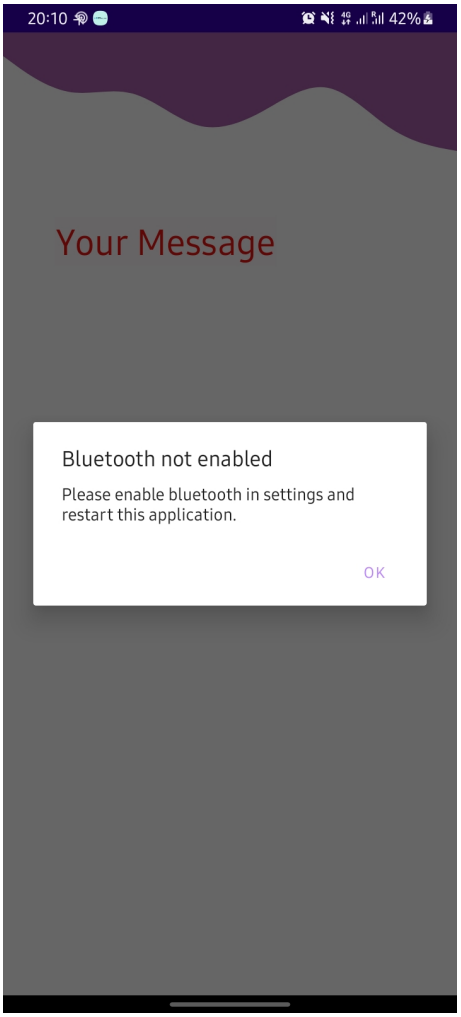
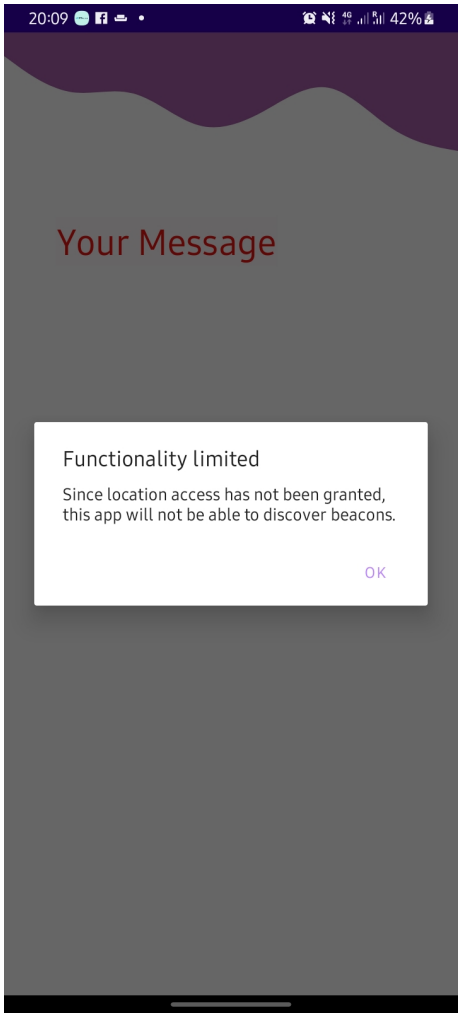


Figure B.2: Beacons

B.0.3 Other Images from the Application



(a) Bluetooth Alert



(b) Location Alert

Figure B.3: Alerts

Appendix C

Code - Login Register

```
1
2  <?xml version="1.0" encoding="utf-8"?>
3  <androidx.constraintlayout.widget.ConstraintLayout
4      xmlns:android="http://schemas.android.com/apk/res/android"
5      xmlns:app="http://schemas.android.com/apk/res-auto"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      xmlns:tools="http://schemas.android.com/tools"
9      tools:context=".ui.Register.RegisterActivity">
10
11     <View
12         android:id="@+id/topbar"
13         android:layout_width="0dp"
14         android:layout_height="150dp"
15         android:background="@color/colorPrimary"
16         app:layout_constraintEnd_toEndOf="parent"
17         app:layout_constraintHorizontal_bias="0.0"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toTopOf="parent" />
20
21
22     <View
```

```
        android:id="@+id/wave"
        android:layout_width="0dp"
        android:layout_height="100dp"
        android:background="@drawable/wave1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/topbar" />



    <EditText
        android:id="@+id/registerednumber"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/colorPrimary"
        android:hint="Registered Number School"
        android:inputType="number"
        android:paddingStart="10dp"
        android:paddingTop="20dp"
        android:paddingBottom="10dp"
        android:textColorHint="@color/colorPrimary"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.353"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.343"
        app:layout_constraintWidth_percent="0.8"/>

    <EditText
        android:id="@+id/username"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/colorPrimary"
```

```
        android:hint="Username"
        android:inputType="textPersonName"
        android:paddingStart="10dp"
        android:paddingTop="20dp"
        android:paddingBottom="10dp"
        android:textColorHint="@color/colorPrimary"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.353"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/registerednumber"
        app:layout_constraintVertical_bias="0.0"
        app:layout_constraintWidth_percent="0.8" />


    <EditText
        android:id="@+id/email"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/colorPrimary"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:paddingStart="10dp"
        android:paddingTop="20dp"
        android:paddingBottom="10dp"
        android:textColorHint="@color/colorPrimary"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.353"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/username"
        app:layout_constraintVertical_bias="0.0"
        app:layout_constraintWidth_percent="0.8" />


```

```
93      <EditText
94          android:id="@+id/password"
95          android:layout_width="0dp"
96          android:layout_height="wrap_content"
97          android:backgroundTint="@color/colorPrimary"
98          android:hint="Password"
99          android:inputType="textPassword"
100         android:paddingStart="10dp"
101         android:paddingTop="20dp"
102         android:paddingBottom="10dp"
103         android:textColorHint="@color/colorPrimary"
104         app:layout_constraintBottom_toBottomOf="parent"
105         app:layout_constraintEnd_toEndOf="parent"
106         app:layout_constraintHorizontal_bias="0.353"
107         app:layout_constraintStart_toStartOf="parent"
108         app:layout_constraintTop_toBottomOf="@+id/email"
109         app:layout_constraintVertical_bias="0.0"
110         app:layout_constraintWidth_percent="0.8" />
111
112     <EditText
113         android:id="@+id/password2"
114         android:layout_width="0dp"
115         android:layout_height="wrap_content"
116         android:backgroundTint="@color/colorPrimary"
117         android:hint="@string/confirm_password"
118         android:inputType="textPassword"
119         android:paddingStart="10dp"
120         android:paddingTop="20dp"
121         android:paddingBottom="10dp"
122         android:textColorHint="@color/colorPrimary"
123         app:layout_constraintBottom_toBottomOf="parent"
124         app:layout_constraintEnd_toEndOf="parent"
125         app:layout_constraintHorizontal_bias="0.353"
126         app:layout_constraintStart_toStartOf="parent"
127         app:layout_constraintTop_toBottomOf="@+id/password"
```

```
128         app:layout_constraintVertical_bias="0.0"
129         app:layout_constraintWidth_percent="0.8" />
130
131     <Button
132         android:id="@+id/signup"
133         android:layout_width="0dp"
134         android:layout_height="wrap_content"
135         android:layout_marginTop="88dp"
136         android:background="@drawable/register_loginbutton"
137         android:text="@string/sign_up"
138         android:textSize="16sp"
139         android:textStyle="bold"
140         android:textColor="#FFFFFF"
141         app:layout_constraintEnd_toEndOf="parent"
142         app:layout_constraintStart_toStartOf="parent"
143         app:layout_constraintTop_toBottomOf="@+id/password"
144         app:layout_constraintWidth_percent=".4" />
145
146     <TextView
147         android:id="@+id/accountalready"
148         android:layout_width="wrap_content"
149         android:layout_height="wrap_content"
150         android:text="@string/already_have_an_account"
151         android:textColor="#D51616"
152         android:textSize="24sp"
153         app:layout_constraintBottom_toTopOf="@+id/loginbutton"
154         app:layout_constraintEnd_toEndOf="parent"
155         app:layout_constraintStart_toStartOf="parent"
156         app:layout_constraintTop_toBottomOf="@+id/signup"
157         app:layout_constraintVertical_bias="0.274" />
158
159     <Button
160         android:id="@+id/loginbutton"
161         android:layout_width="wrap_content"
162         android:layout_height="wrap_content"
```

```
        android:layout_marginBottom="16dp"
        android:background="#FF7F7F"
        android:text="@string/login"
        android:textColor="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Listing C.1: activity_login.xml

```

public class LoginActivity extends AppCompatActivity {
    private Button registerButton;
    Button btnLogin;
    EditText edEmail, edPassword;
    Intent intent;
    SharedPrefManager sharedPrefManager;


    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        btnLogin = findViewById(R.id.loginbutton);
        edEmail= findViewById(R.id.emaillogin);
        edPassword= findViewById(R.id.passwordlogin);
        registerButton = (Button) findViewById(R.id.register);
        sharedPrefManager=new SharedPrefManager(getApplicationContext())
    ;


```

```
22
23
24
25      btnLogin.setOnClickListener(new View.OnClickListener() {
26          @Override
27          public void onClick(View view) {
28              if (TextUtils.isEmpty(edEmail.getText().toString()) ||
    TextUtils.isEmpty(edPassword.getText().toString())) {
29                  String message = "All input required...";
30                  Toast.makeText(LoginActivity.this, message, Toast.
    LENGTH_LONG).show();
31
32              } else {
33                  LoginRequest loginRequest = new LoginRequest();
34                  loginRequest.setUsername(edEmail.getText().toString
    () );
35                  loginRequest.setPassword(edPassword.getText().
    toString());
36
37                  loginUser(loginRequest);
38                  String message1 = "Login Successfull...";
39                  Toast.makeText(LoginActivity.this, message1, Toast.
    LENGTH_LONG).show();
40
41
42              }
43          }
44      });
45
46
47      registerButton.setOnClickListener(new View.OnClickListener() {
48          @Override
49          public void onClick(View view) {
50              Register();
51          }
```

```
52
53
54          });
55      }
56      /*@Override
57      protected void onStart() {
58          super.onStart();
59
60          if (SharedPrefManager.getInstance(this).isLoggedIn()) {
61              Intent intent = new Intent(this, MainActivity.class);
62              intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.
    FLAG_ACTIVITY_CLEAR_TASK);
63              startActivity(intent);
64          }
65      }*/
66      private void Register() {
67          intent = new Intent(this, RegisterActivity.class);
68          startActivity(intent);
69      }
70      public void loginUser(LoginRequest loginRequest) {
71          Call<LoginResponse> loginResponseCall = ApiClient.getServices().
    loginUser(loginRequest);
72          loginResponseCall.enqueue(new Callback<LoginResponse>() {
73              @Override
74              public void onResponse(Call<LoginResponse> call, Response<
    LoginResponse> response) {
75                  LoginResponse loginResponse = response.body();
76                  if(response.isSuccessful()){
77
78                    //  assert loginResponse != null;
79                      assert loginResponse != null;
80                      SharedPrefManager.getInstance(LoginActivity.this)
81                              .saveUser(loginResponse.getUser());
82                      Intent intent = new Intent(LoginActivity.this,
    MainActivity.class);
```

```
                    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                    startActivity(intent);

                    finish();


                }else{
                    String message = "An error occurred please try again
 later...";
                    Toast.makeText(LoginActivity.this,message,Toast.
LENGTH_LONG).show();


                }
            }



            @Override
            public void onFailure(Call<LoginResponse> call, Throwable t)
 {
                String message = t.getLocalizedMessage();
                Toast.makeText(LoginActivity.this,message,Toast.
LENGTH_LONG).show();
            }
        });


    }
    protected void onStart(){
        super.onStart();

        if(sharedPrefManager.isLoggedIn()){
```

```
113             Intent intent = new Intent(LoginActivity.this, MainActivity.
    class);
114             intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK| Intent.
    FLAG_ACTIVITY_CLEAR_TASK);
115             startActivity(intent);
116
117         }
118     }
119
120 }
```

Listing C.2: LoginActivity.java

```
1
2 <?xml version="1.0" encoding="utf-8"?>
3 <androidx.constraintlayout.widget.ConstraintLayout
4     xmlns:android="http://schemas.android.com/apk/res/android"
5     xmlns:app="http://schemas.android.com/apk/res-auto"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     xmlns:tools="http://schemas.android.com/tools"
9     tools:context=".ui.Register.RegisterActivity">
10
11     <View
12         android:id="@+id/topbar"
13         android:layout_width="0dp"
14         android:layout_height="150dp"
15         android:background="@color/colorPrimary"
16         app:layout_constraintEnd_toEndOf="parent"
17         app:layout_constraintHorizontal_bias="0.0"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toTopOf="parent" />
20
21
22     <View
23         android:id="@+id/wave"
```

```
        android:layout_width="0dp"
        android:layout_height="100dp"
        android:background="@drawable/wave1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/topbar" />



    <EditText
        android:id="@+id/registerednumber"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/colorPrimary"
        android:hint="Registered Number School"
        android:inputType="number"
        android:paddingStart="10dp"
        android:paddingTop="20dp"
        android:paddingBottom="10dp"
        android:textColorHint="@color/colorPrimary"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.353"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.343"
        app:layout_constraintWidth_percent="0.8"/>

    <EditText
        android:id="@+id/username"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/colorPrimary"
        android:hint="Username"
```

```
59          android:inputType="textPersonName"
60          android:paddingStart="10dp"
61          android:paddingTop="20dp"
62          android:paddingBottom="10dp"
63          android:textColorHint="@color/colorPrimary"
64          app:layout_constraintBottom_toBottomOf="parent"
65          app:layout_constraintEnd_toEndOf="parent"
66          app:layout_constraintHorizontal_bias="0.353"
67          app:layout_constraintStart_toStartOf="parent"
68          app:layout_constraintTop_toBottomOf="@+id/registerednumber"
69          app:layout_constraintVertical_bias="0.0"
70          app:layout_constraintWidth_percent="0.8" />
71
72
73      <EditText
74          android:id="@+id/email"
75          android:layout_width="0dp"
76          android:layout_height="wrap_content"
77          android:backgroundTint="@color/colorPrimary"
78          android:hint="Email"
79          android:inputType="textEmailAddress"
80          android:paddingStart="10dp"
81          android:paddingTop="20dp"
82          android:paddingBottom="10dp"
83          android:textColorHint="@color/colorPrimary"
84          app:layout_constraintBottom_toBottomOf="parent"
85          app:layout_constraintEnd_toEndOf="parent"
86          app:layout_constraintHorizontal_bias="0.353"
87          app:layout_constraintStart_toStartOf="parent"
88          app:layout_constraintTop_toBottomOf="@+id/username"
89          app:layout_constraintVertical_bias="0.0"
90          app:layout_constraintWidth_percent="0.8" />
91
92
93      <EditText
```

```
94          android:id="@+id/password"
95          android:layout_width="0dp"
96          android:layout_height="wrap_content"
97          android:backgroundTint="@color/colorPrimary"
98          android:hint="Password"
99          android:inputType="textPassword"
100         android:paddingStart="10dp"
101         android:paddingTop="20dp"
102         android:paddingBottom="10dp"
103         android:textColorHint="@color/colorPrimary"
104         app:layout_constraintBottom_toBottomOf="parent"
105         app:layout_constraintEnd_toEndOf="parent"
106         app:layout_constraintHorizontal_bias="0.353"
107         app:layout_constraintStart_toStartOf="parent"
108         app:layout_constraintTop_toBottomOf="@+id/email"
109         app:layout_constraintVertical_bias="0.0"
110         app:layout_constraintWidth_percent="0.8" />
111
112     <EditText
113         android:id="@+id/password2"
114         android:layout_width="0dp"
115         android:layout_height="wrap_content"
116         android:backgroundTint="@color/colorPrimary"
117         android:hint="@string/confirm_password"
118         android:inputType="textPassword"
119         android:paddingStart="10dp"
120         android:paddingTop="20dp"
121         android:paddingBottom="10dp"
122         android:textColorHint="@color/colorPrimary"
123         app:layout_constraintBottom_toBottomOf="parent"
124         app:layout_constraintEnd_toEndOf="parent"
125         app:layout_constraintHorizontal_bias="0.353"
126         app:layout_constraintStart_toStartOf="parent"
127         app:layout_constraintTop_toBottomOf="@+id/password"
128         app:layout_constraintVertical_bias="0.0"
```

```
129         app:layout_constraintWidth_percent="0.8" />
130
131     <Button
132         android:id="@+id/signup"
133         android:layout_width="0dp"
134         android:layout_height="wrap_content"
135         android:layout_marginTop="88dp"
136         android:background="@drawable/register_loginbutton"
137         android:text="@string/sign_up"
138         android:textSize="16sp"
139         android:textStyle="bold"
140         android:textColor="#FFFFFF"
141         app:layout_constraintEnd_toEndOf="parent"
142         app:layout_constraintStart_toStartOf="parent"
143         app:layout_constraintTop_toBottomOf="@+id/password"
144         app:layout_constraintWidth_percent=".4" />
145
146     <TextView
147         android:id="@+id/accountalready"
148         android:layout_width="wrap_content"
149         android:layout_height="wrap_content"
150         android:text="@string/already_have_an_account"
151         android:textColor="#D51616"
152         android:textSize="24sp"
153         app:layout_constraintBottom_toTopOf="@+id/loginbutton"
154         app:layout_constraintEnd_toEndOf="parent"
155         app:layout_constraintStart_toStartOf="parent"
156         app:layout_constraintTop_toBottomOf="@+id/signup"
157         app:layout_constraintVertical_bias="0.274" />
158
159     <Button
160         android:id="@+id/loginbutton"
161         android:layout_width="wrap_content"
162         android:layout_height="wrap_content"
163         android:layout_marginBottom="16dp"
```

```
        android:background="#FF7F7F"
        android:text="@string/login"
        android:textColor="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Listing C.3: activity_register.xml

```

public class RegisterActivity extends AppCompatActivity {
    Button btnSignUp;
   EditText  regNum, edUsername, edEmail, edPassword, password2;
   Button btnLogin;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        btnSignUp =findViewById(R.id.signup);
        regNum =findViewById(R.id.registerednumber);
        edUsername =findViewById(R.id.username);
        edEmail=findViewById(R.id.email);
        edPassword=findViewById(R.id.password);
        password2=findViewById(R.id.password2);
        btnLogin=findViewById(R.id.loginbutton);

        btnSignUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
21              if(TextUtils.isEmpty(edEmail.getText().toString()) ||
    TextUtils.isEmpty(edUsername.getText().toString()) || TextUtils.
    isEmpty(edPassword.getText().toString()) || TextUtils.isEmpty(
    password2.getText().toString()) || TextUtils.isEmpty(regNum.getText()
    .toString())){
22                  String message = "All input required...";
23                  Toast.makeText(RegisterActivity.this,message,Toast.
    LENGTH_LONG).show();
24
25              }else {
26                  RegisterRequest registerRequest = new
    RegisterRequest();
27                  registerRequest.setEmail(edEmail.getText().toString
    ());
28                  registerRequest.setPassword((edPassword.getText().
    toString()));
29                  registerRequest.setPassword2((password2.getText().
    toString()));
30                  registerRequest.setUsername((edUsername.getText().
    toString()));
31                  registerRequest.setYourNumberinSchool(regNum.getText
    ().toString());
32                  registerUser(registerRequest);
33                  String message = "User registered successfully...";
34                  Toast.makeText(RegisterActivity.this, message, Toast
    .LENGTH_LONG).show();
35              }
36
37          }
38      });
39
40      btnLogin.setOnClickListener(new View.OnClickListener() {
41          @Override
42          public void onClick(View view) {
43              Login();
```

```
44              }
45
46
47          });
48
49
50
51      }
52      private void Login() {
53          Intent intent = new Intent(this, LoginActivity.class);
54          startActivity(intent);
55      }
56
57
58
59  public  void registerUser(RegisterRequest registerRequest){
60          Call<RegisterResponse> registerResponseCall = ApiClient.
    getServices().registerUsers(registerRequest);
61          registerResponseCall.enqueue(new Callback<RegisterResponse>() {
62              @Override
63              public void onResponse(Call<RegisterResponse> call, Response
    <RegisterResponse> response) {
64                  if(response.isSuccessful()){
65                      String message = "Successful...";
66                      Toast.makeText(RegisterActivity.this,message,Toast.
    LENGTH_LONG).show();
67
68                      startActivity(new Intent(RegisterActivity.this,
    LoginActivity.class));
69                      finish();
70
71
72
73                  }else{
74
```

```
75                    String message = "An error occurred please try again
      later...";
76                    Toast.makeText(RegisterActivity.this,message,Toast.
      LENGTH_LONG).show();
77
78                }
79            }
80
81            @Override
82            public void onFailure(Call<RegisterResponse> call, Throwable
       t) {
83
84                String message = t.getLocalizedMessage();
85                Toast.makeText(RegisterActivity.this,message,Toast.
      LENGTH_LONG).show();
86            }
87        });
88
89    }
90
91 }
```

Listing C.4: RegisterActivity.java

Appendix D

Code - Main Activities like detecting beacons and getting informations from cloud

```
1
2 package com.example.ipbeacons_mobile_app.ui.BeaconDetection;
3
4 import android.app.Application;
5 import android.app.Notification;
6 import android.app.NotificationChannel;
7 import android.app.NotificationManager;
8 import android.app.PendingIntent;
9 import android.app.TaskStackBuilder;
10 import android.content.Context;
11 import android.content.Intent;
12 import android.os.Build;
13 import android.util.Log;
14 import android.widget.Toast;
15
16 import com.example.ipbeacons_mobile_app.R;
17 import com.example.ipbeacons_mobile_app.ui.Home.HomeActivity;
```

```
18 import com.example.ipbeacons_mobile_app.ui.Login.LoginActivity;
19 import com.example.ipbeacons_mobile_app.ui.Storage.SharedPrefManager;
20
21 import org.altbeacon.beacon.BeaconManager;
22 import org.altbeacon.beacon.BeaconParser;
23 import org.altbeacon.beacon.MonitorNotifier;
24 import org.altbeacon.beacon.Region;
25
26 public class BeaconDetectionActivity extends Application implements
       MonitorNotifier {
27     private static final String TAG = "BeaconDetection";
28     public static final Region wildcardRegion = new Region("
       wildcardRegion", null, null, null);
29     public static boolean insideRegion = false;
30     SharedPrefManager sharedPrefManager;
31
32
33     public  void onCreate(){
34         super.onCreate();
35         sharedPrefManager=new SharedPrefManager(getApplicationContext())
       ;
36
37         BeaconManager beaconManager = org.altbeacon.beacon.BeaconManager
       .getInstanceForApplication(this);
38
39         // beaconManager.getBeaconParsers().add(new BeaconParser().
       setBeaconLayout("m:0-3=4c000215,i:4-19,i:20-21,i:22-23,p:24-24"));
40         beaconManager.getBeaconParsers().clear();
41         beaconManager.getBeaconParsers().add(new BeaconParser().
       setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24"));
42          beaconManager.getBeaconParsers().add(new BeaconParser().
       setBeaconLayout("m:0-3=4c000215,i:4-19,i:20-21,i:22-23,p:24-24"));
43
44         beaconManager.setDebug(true);
45
```

```
46          Notification.Builder builder = new Notification.Builder(this);
47          builder.setSmallIcon(R.mipmap.ic_launcher_round);
48          builder.setContentTitle("Scanning for Beacons");
49          Intent intent = new Intent(this, HomeActivity.class);
50          PendingIntent pendingIntent = PendingIntent.getActivity(
51                  this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT
52          );
53          builder.setContentIntent(pendingIntent);
54          if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
55              NotificationChannel channel = new NotificationChannel("
    Notification ChannelID",
56                      "NotificationName", NotificationManager.
    IMPORTANCE_DEFAULT);
57              channel.setDescription("Notification Channel Description");
58              NotificationManager notificationManager = (
    NotificationManager) getSystemService(
59                      Context.NOTIFICATION_SERVICE);
60              notificationManager.createNotificationChannel(channel);
61              builder.setChannelId(channel.getId());
62          }
63          beaconManager.enableForegroundServiceScanning(builder.build(),
    456);
64
65          beaconManager.setEnableScheduledScanJobs(false);
66          beaconManager.setBackgroundBetweenScanPeriod(0);
67          beaconManager.setBackgroundScanPeriod(1100);
68
69          Log.d(TAG, "Background Monitoring");
70          beaconManager.addMonitorNotifier(this);
71
72          for (Region region: beaconManager.getMonitoredRegions()) {
73              beaconManager.stopMonitoring(region);
74          }
75
76          beaconManager.startMonitoring(wildcardRegion);
```

```
77
78          }
79
80
81      @Override
82      public void didEnterRegion(Region region) {
83          Log.d(TAG, "did enter region.");
84          insideRegion = true;
85          // Send a notification to the user whenever a Beacon
86          // matching a Region (defined above) are first seen.
87          Log.d(TAG, "Sending notification.");
88          sendNotification();
89
90      }
91
92
93      @Override
94      public void didExitRegion(Region region) {
95          insideRegion = false;
96
97
98      }
99
100     @Override
101     public void didDetermineStateForRegion(int state, Region region) {
102
103
104     }
105
106     private void sendNotification() {
107         NotificationManager notificationManager =
108                 (NotificationManager) this.getSystemService(Context.
    NOTIFICATION_SERVICE);
109         Notification.Builder builder;
110         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
```

```
111             NotificationChannel channel = new NotificationChannel("
    IPBeacon Notifications",
112                     "IPBeacon Notifications", NotificationManager.
    IMPORTANCE_HIGH);
113             channel.enableLights(true);
114             channel.enableVibration(true);
115             channel.setLockscreenVisibility(Notification.
    VISIBILITY_PUBLIC);
116             notificationManager.createNotificationChannel(channel);
117             builder = new Notification.Builder(this, channel.getId());
118         }
119         else {
120             builder = new Notification.Builder(this);
121             builder.setPriority(Notification.PRIORITY_HIGH);
122         }
123         TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
124         stackBuilder.addNextIntent(new Intent(this, HomeActivity.class))
    ;
125         PendingIntent resultPendingIntent =
126                 stackBuilder.getPendingIntent(
127                         0,
128                         PendingIntent.FLAG_UPDATE_CURRENT
129                 );
130
131         TaskStackBuilder stackBuilder1 = TaskStackBuilder.create(this);
132         stackBuilder1.addNextIntent(new Intent(this, LoginActivity.class
    ));
133         PendingIntent resultPending = stackBuilder1.getPendingIntent(0,
    PendingIntent.FLAG_UPDATE_CURRENT);
134         builder.setSmallIcon(R.mipmap.ic_launcher_round);
135         builder.setContentTitle("Beacon Detected");
136         builder.setContentText("You have important notification incoming
    . Tap here to view");
137         if(sharedPrefManager.isLoggedIn()){
138             builder.setContentIntent(resultPendingIntent);
```

```
139
140         }
141         else{
142             builder.setContentIntent(resultPending);
143
144         }
145
146         notificationManager.notify(1, builder.build());
147     }
148 }
```

Listing D.1: BeaconDetectionActivity.java

```
1  package com.example.ipbeacons_mobile_app;
2
3  import androidx.appcompat.app.AppCompatActivity;
4
5  import android.annotation.SuppressLint;
6  import android.content.Context;
7  import android.content.Intent;
8  import android.os.Bundle;
9  import android.util.Log;
10 import android.view.View;
11 import android.widget.Button;
12 import android.widget.EditText;
13 import android.widget.RadioButton;
14 import android.widget.TextView;
15 import android.widget.Toast;
16
17
18 import com.example.ipbeacons_mobile_app.api.LoginResponse;
19 import com.example.ipbeacons_mobile_app.api.User;
20 import com.example.ipbeacons_mobile_app.ui.Home.HomeActivity;
21 import com.example.ipbeacons_mobile_app.ui.Login.LoginActivity;
22 import com.example.ipbeacons_mobile_app.ui.Register.RegisterActivity;
23 import com.example.ipbeacons_mobile_app.ui.Storage.SharedPrefManager;
```

```
24 import com.google.android.material.textfield.TextInputEditText;
25
26 public class MainActivity extends AppCompatActivity {
27     private Button LogOut;
28     private Button gotohome;
29     // private Button update;
30     TextView welcomeUser;
31     LoginResponse loginResponse;
32     User userObject;
33     TextInputEditText user;
34     TextInputEditText number;
35     TextInputEditText email;
36     SharedPrefManager sharedPrefManager;
37
38
39     @Override
40     protected void onCreate(Bundle savedInstanceState) {
41         super.onCreate(savedInstanceState);
42         setContentView(R.layout.activity_landing);
43         sharedPrefManager = new SharedPrefManager(getApplicationContext
    ());
44
45
46         user = findViewById(R.id.name);
47         number = findViewById(R.id.num);
48         email = findViewById(R.id.mail);
49         welcomeUser = findViewById(R.id.welcomeUser);
50         sharedPrefManager = new SharedPrefManager(getApplicationContext
    ());
51
52         Intent intent = getIntent();
53         //  if(intent.getExtras() != null){
54         //  sharedPrefManager = (SharedPrefManager) intent.
    getSerializableExtra("data");
55         user.setText(sharedPrefManager.getUser().getUsername());
```

```
56          number.setText(sharedPrefManager.getUser().getYourNumberinSchool
    ());
57          email.setText(sharedPrefManager.getUser().getEmail());
58          welcomeUser.setText(sharedPrefManager.getUser().getUsername());
59
60
61          //  }
62          LogOut = (Button) findViewById(R.id.logout);
63          gotohome = (Button) findViewById(R.id.ipbeacon);
64          //update = (Button) findViewById(R.id.update);
65
66          gotohome.setOnClickListener(new View.OnClickListener() {
67              @Override
68              public void onClick(View view) {
69                  Home();
70              }
71          });
72         /* update.setOnClickListener(new View.OnClickListener() {
73              @Override
74              public void onClick(View view) {
75                  Update();
76              }
77          });*/
78
79          LogOut.setOnClickListener(new View.OnClickListener() {
80              @Override
81              public void onClick(View view) {
82                  logout();
83              }
84          });
85      }
86
87      private void logout() {
88          sharedPrefManager.clear();
```

```
89          Intent intent = new Intent(MainActivity.this, LoginActivity.
    class);
90          intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.
    FLAG_ACTIVITY_CLEAR_TASK);
91
92          startActivity(intent);
93          Toast.makeText(this, "LogOut successful", Toast.LENGTH_LONG).
    show();
94      }
95
96      public void Home() {
97          Intent intent = new Intent(this, HomeActivity.class);
98          startActivity(intent);
99      }
100 }
```

Listing D.2: MainActivity.java

```
1  package com.example.ipbeacons_mobile_app.ui.Home;
2
3  import android.Manifest;
4  import android.annotation.SuppressLint;
5  import android.annotation.TargetApi;
6  import android.app.Activity;
7  import android.app.AlertDialog;
8  import android.content.DialogInterface;
9  import android.content.pm.PackageManager;
10 import android.os.Build;
11 import android.os.Bundle;
12 import android.util.Log;
13 import android.view.LayoutInflater;
14 import android.view.View;
15 import android.widget.Adapter;
16 import android.widget.LinearLayout;
17 import android.widget.ProgressBar;
18 import android.widget.TextView;
```

```
19 import android.widget.Toast;
20
21 import androidx.recyclerview.widget.LinearLayoutManager;
22 import androidx.recyclerview.widget.RecyclerView;
23
24 import com.example.ipbeacons_mobile_app.R;
25 import com.example.ipbeacons_mobile_app.api.ApiClient;
26 import com.example.ipbeacons_mobile_app.api.Info;
27 import com.example.ipbeacons_mobile_app.api.InfoAdapter;
28 import com.example.ipbeacons_mobile_app.ui.BeaconDetection.
    BeaconDetectionActivity;
29 import com.pluscubed.recyclerfastscroll.RecyclerFastScroller;
30
31 import org.altbeacon.beacon.Beacon;
32 import org.altbeacon.beacon.BeaconManager;
33 import org.altbeacon.beacon.MonitorNotifier;
34 import org.altbeacon.beacon.RangeNotifier;
35 import org.altbeacon.beacon.Region;
36
37 import java.util.ArrayList;
38 import java.util.Collection;
39 import java.util.List;
40
41 import retrofit2.Call;
42 import retrofit2.Callback;
43 import retrofit2.Response;
44
45 public class HomeActivity extends Activity implements MonitorNotifier {
46     protected static final String TAG = "HomeActivity";
47     private static final int PERMISSION_REQUEST_FINE_LOCATION = 1;
48     private static final int PERMISSION_REQUEST_BACKGROUND_LOCATION = 2;
49     private BeaconManager beaconManager = BeaconManager.
    getInstanceForApplication(this);
50
51
```

```
52      RecyclerView recyclerView;
53      ProgressBar progressBar;
54      LinearLayoutManager linearLayoutManager;
55      InfoAdapter adapter;
56      List<Info>infoList=new ArrayList<>();
57
58
59      @Override
60      protected void onCreate(Bundle savedInstanceState) {
61
62          super.onCreate(savedInstanceState);
63
64
65
66          verifyBluetooth();
67          requestPermissions();
68          BeaconManager.getInstanceForApplication(this).addMonitorNotifier
    (this);
69      }
70
71
72      @Override
73      protected void onResume() {
74          super.onResume();
75
76          setContentView(R.layout.activity_home);
77          recyclerView = findViewById(R.id.recycleview);
78          progressBar = findViewById(R.id.progress);
79          linearLayoutManager=new LinearLayoutManager(this);
80          recyclerView.setLayoutManager(linearLayoutManager);
81          adapter = new InfoAdapter(infoList,getApplicationContext());
82          recyclerView.setAdapter(adapter);
83          progressBar.setVisibility(View.VISIBLE);
84
85
```

```
86
87          RangeNotifier rangeNotifier = new RangeNotifier() {
88              @Override
89              public void didRangeBeaconsInRegion(Collection<Beacon>
    beacons, Region region) {
90                  if(beacons.size()>0){
91                      for(Beacon b:beacons){
92                          Call<List<Info>> infoCall = ApiClient.getApi().
    getInfos(b.getId1().toString());
93
94                          infoCall.enqueue(new Callback<List<Info>>() {
95                              @SuppressLint("SetTextI18n")
96                              @Override
97                              public void onResponse(Call<List<Info>> call
    , Response<List<Info>> response) {
98                                  if(response.isSuccessful() && response.
    body()!=null){
99
100                                     infoList.clear();
101                                     infoList.addAll(response.body());
102                                     adapter.notifyDataSetChanged();
103                                     progressBar.setVisibility(View.GONE)
    ;
104                                 }
105                                 /*  if(!response.isSuccessful()){
106                                     textView.setText("Code: "+ response.
    code());
107                                     return;
108                                 }
109                                 List<Info> infos =response.body();
110                                 for(Info beacon : infos){
111                                     String content ="";
112                                  *//*   content +="Beacon_ID: " + beacon
    .getBeaconID() + "\n";
```

```
113                                 content +="Information_ID: " +
    beacon.getInforamtionID() + "\n";
114                                 content +="Datetime: " + beacon.
    getDatetime() + "\n";*//*
115                                 content +="Information_Text: " +
    beacon.getInformationText() + "\n";
116
117                                 //
118                                 //     content +="NameOfBeacon: " +
    beacon.getBeaconName() + "\n";
119
120                                 textView.append(content);
121
122                             }*/
123                         }
124
125
126                         @Override
127                         public void onFailure(Call<List<Info>> call,
     Throwable t) {
128                             progressBar.setVisibility(View.GONE);
129                             Toast.makeText(HomeActivity.this,"Error:
     "+ t.getMessage(),Toast.LENGTH_SHORT).show();
130                         }
131                         });
132
133
134                   /*  if(b.getId1().toString().equals(beaconID1))
135                         Display("specific beacon found");*/
136                 }
137             }
138             else if(beacons.size()==0){
139                 progressBar.setVisibility(View.GONE);
140             }
141                 /*for(Beacon b:beacons){
```

```
142                         if(b.getId1().toString().equals(beaconIDl))
143                         Display("specific beacon found");
144                     }*/
145
146             }
147         };
148
149         beaconManager.addRangeNotifier(rangeNotifier);
150         beaconManager.startRangingBeacons(BeaconDetectionActivity.
    wildcardRegion);
151
152
153     }
154
155     @Override
156     public void didEnterRegion(Region region) {
157
158     }
159
160     @Override
161     public void didExitRegion(Region region) {
162
163     }
164
165     @Override
166     public void didDetermineStateForRegion(int state, Region region) {
167
168     }
169
170     private void verifyBluetooth() {
171         try {
172             if (!BeaconManager.getInstanceForApplication(this).
    checkAvailability()) {
173                 final AlertDialog.Builder builder = new AlertDialog.
    Builder(this);
```

```
174                 builder.setTitle("Bluetooth not enabled");
175                 builder.setMessage("Please enable bluetooth in settings
    and restart this application.");
176                 builder.setPositiveButton(android.R.string.ok, null);
177                 builder.setOnDismissListener(new DialogInterface.
    OnDismissListener() {
178                     @Override
179                     public void onDismiss(DialogInterface dialog) {
180                         finishAffinity();
181                     }
182                 });
183                 builder.show();
184             }
185         }
186         catch (RuntimeException e) {
187             final AlertDialog.Builder builder = new AlertDialog.Builder(
    this);
188             builder.setTitle("Bluetooth LE not available");
189             builder.setMessage("Sorry, this device does not support
    Bluetooth LE.");
190             builder.setPositiveButton(android.R.string.ok, null);
191             builder.setOnDismissListener(new DialogInterface.
    OnDismissListener() {
192
193                 @Override
194                 public void onDismiss(DialogInterface dialog) {
195                     finishAffinity();
196                 }
197
198             });
199             builder.show();
200
201         }
202
203     }
```

```
204
205     private void requestPermissions() {
206         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
207             if (this.checkSelfPermission(Manifest.permission.
    ACCESS_FINE_LOCATION)
208                     == PackageManager.PERMISSION_GRANTED) {
209                 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
210                     if (this.checkSelfPermission(Manifest.permission.
    ACCESS_BACKGROUND_LOCATION)
211                             != PackageManager.PERMISSION_GRANTED) {
212                         if (!this.shouldShowRequestPermissionRationale(
    Manifest.permission.ACCESS_BACKGROUND_LOCATION)) {
213                             final AlertDialog.Builder builder = new
    AlertDialog.Builder(this);
214                             builder.setTitle("This app needs background
    location access");
215                             builder.setMessage("Please grant location
    access so this app can detect beacons in the background.");
216                             builder.setPositiveButton(android.R.string.
    ok, null);
217                             builder.setOnDismissListener(new
    DialogInterface.OnDismissListener() {
218
219                                 @TargetApi(23)
220                                 @Override
221                                 public void onDismiss(DialogInterface
    dialog) {
222                                     requestPermissions(new String[]{
    Manifest.permission.ACCESS_BACKGROUND_LOCATION},
223
    PERMISSION_REQUEST_BACKGROUND_LOCATION);
224                                 }
225
226                             });
227                             builder.show();
```

```
228                         }
229                         else {
230                             final AlertDialog.Builder builder = new
    AlertDialog.Builder(this);
231                             builder.setTitle("Functionality limited");
232                             builder.setMessage("Since background
    location access has not been granted, this app will not be able to
    discover beacons in the background.  Please go to Settings ->
    Applications -> Permissions and grant background location access to
    this app.");
233                             builder.setPositiveButton(android.R.string.
    ok, null);
234                             builder.setOnDismissListener(new
    DialogInterface.OnDismissListener() {
235
236                                 @Override
237                                 public void onDismiss(DialogInterface
    dialog) {
238                                 }
239
240                             });
241                             builder.show();
242                         }
243                     }
244                 }
245             } else {
246                 if (!this.shouldShowRequestPermissionRationale(Manifest.
    permission.ACCESS_FINE_LOCATION)) {
247                     requestPermissions(new String[]{Manifest.permission.
    ACCESS_FINE_LOCATION,
248                                     Manifest.permission.
    ACCESS_BACKGROUND_LOCATION},
249                             PERMISSION_REQUEST_FINE_LOCATION);
250                 }
251                 else {
```

```
252                     final AlertDialog.Builder builder = new AlertDialog.
    Builder(this);
253                     builder.setTitle("Functionality limited");
254                     builder.setMessage("Since location access has not
    been granted, this app will not be able to discover beacons.  Please
    go to Settings -> Applications -> Permissions and grant location
    access to this app.");
255                     builder.setPositiveButton(android.R.string.ok, null)
    ;
256                     builder.setOnDismissListener(new DialogInterface.
    OnDismissListener() {
257
258                         @Override
259                         public void onDismiss(DialogInterface dialog) {
260                         }
261
262                     });
263                     builder.show();
264                 }
265
266             }
267         }
268     }
269
270     public void onRequestPermissionsResult(int requestCode,
271                                            String permissions[], int[]
    grantResults) {
272         switch (requestCode) {
273             case PERMISSION_REQUEST_FINE_LOCATION: {
274                 if (grantResults[0] == PackageManager.PERMISSION_GRANTED
    ) {
275                     Log.d(TAG, "fine location permission granted");
276                 } else {
277                     final AlertDialog.Builder builder = new AlertDialog.
    Builder(this);
```

```
278                     builder.setTitle("Functionality limited");
279                     builder.setMessage("Since location access has not
    been granted, this app will not be able to discover beacons.");
280                     builder.setPositiveButton(android.R.string.ok, null)
    ;
281                     builder.setOnDismissListener(new DialogInterface.
    OnDismissListener() {
282 
283                         @Override
284                         public void onDismiss(DialogInterface dialog) {
285                         }
286 
287                     });
288                     builder.show();
289                 }
290                 return;
291             }
292             case PERMISSION_REQUEST_BACKGROUND_LOCATION: {
293                 if (grantResults[0] == PackageManager.PERMISSION_GRANTED
    ) {
294                     Log.d(TAG, "background location permission granted")
    ;
295                 } else {
296                     final AlertDialog.Builder builder = new AlertDialog.
    Builder(this);
297                     builder.setTitle("Functionality limited");
298                     builder.setMessage("Since background location access
     has not been granted, this app will not be able to discover beacons
    when in the background.");
299                     builder.setPositiveButton(android.R.string.ok, null)
    ;
300                     builder.setOnDismissListener(new DialogInterface.
    OnDismissListener() {
301 
302                         @Override
```

```
                        public void onDismiss(DialogInterface dialog) {
                        }

                    });
                    builder.show();
                }
                return;
            }
        }
    }


    private String notificationText = "";
    private void Display(String line) {
        notificationText += line+"\n";
        runOnUiThread(new Runnable() {
            public void run() {
                TextView textView = (TextView) HomeActivity.this
                        .findViewById(R.id.nonotification);
                textView.setText(notificationText);
            }
        });
    }
}
```

Listing D.3: HomeActivity.java

```

package com.example.ipbeacons_mobile_app.api;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
```

```
10
11 import androidx.annotation.NonNull;
12 import androidx.recyclerview.widget.RecyclerView;
13
14 import com.example.ipbeacons_mobile_app.R;
15 import com.example.ipbeacons_mobile_app.ui.Info_Page.InfopageActivity;
16
17 import org.w3c.dom.Text;
18
19 import java.util.List;
20
21 public class InfoAdapter extends RecyclerView.Adapter<InfoAdapter.
   ViewHolder> {
22
23     private List<Info> infoList;
24     Context context;
25
26     public InfoAdapter(List<Info> infoList, Context context) {
27         this.infoList = infoList;
28         this.context=context;
29     }
30
31
32
33     @NonNull
34     @Override
35     public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
   viewType) {
36         View view = LayoutInflater.from(parent.getContext())
37                 .inflate(R.layout.list_info, parent, false);
38
39         return new ViewHolder(view);
40     }
41
42     @Override
```

```
43     public void onBindViewHolder(@NonNull ViewHolder holder, int
   position) {
44               final  Info inf= infoList.get(position);
45               holder.body.setText(infoList.get(position).
   getInformationText());
46               holder.body.setOnClickListener(new View.OnClickListener()
   {
47                   @Override
48                   public void onClick(View v) {
49
50                       Intent intent = new Intent(context,
   InfopageActivity.class);
51                       intent.putExtra("information",inf.
   getInformationText());
52                       intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
53                       context.startActivity(intent);
54
55                   }
56               });
57     }
58
59     @Override
60     public int getItemCount() {
61         return infoList.size();
62     }
63
64
65     public class ViewHolder extends RecyclerView.ViewHolder {
66
67       TextView body;
68
69        public ViewHolder(@NonNull View itemView) {
70             super(itemView);
71             body = itemView .findViewById(R.id.information);
72         }
```

```
73
74      }
75  }
```

Listing D.4: InfoAdaptar.java

Appendix E

Code - Retrofit implementation

```
1
2  package com.example.ipbeacons_mobile_app.api;
3
4  import okhttp3.OkHttpClient;
5  import okhttp3.logging.HttpLoggingInterceptor;
6  import retrofit2.Retrofit;
7  import retrofit2.converter.gson.GsonConverterFactory;
8
9  public class ApiClient {
10
11     public static Retrofit getRetrofit(){
12
13         HttpLoggingInterceptor httpLoggingInterceptor = new
       HttpLoggingInterceptor();
14         httpLoggingInterceptor.setLevel(HttpLoggingInterceptor.Level.
       BODY);
15
16         OkHttpClient okHttpClient = new OkHttpClient.Builder().
       addInterceptor(httpLoggingInterceptor).build();
17
18
19         Retrofit retrofit = new Retrofit.Builder()
```

```
20                  .addConverterFactory(GsonConverterFactory.create())
21                  .baseUrl("https://ipbeacon.herokuapp.com/")
22                  .client(okHttpClient)
23                  .build();
24
25          return  retrofit;
26      }
27      public static UserService getServices(){
28          UserService userService = getRetrofit().create(UserService.class
        );
29
30          return userService;
31      }
32      public static JsonPlaceHolderApi getApi(){
33          JsonPlaceHolderApi jsonPlaceHolderApi= getRetrofit().create(
        JsonPlaceHolderApi.class);
34
35          return jsonPlaceHolderApi;
36      }
37  }
```

Listing E.1: ApiClient.java

Appendix F

Code - Django REST API creation

```
1
2  from django.db import models
3
4  # Create your models here.
5  class Beacon(models.Model):
6      beaconID = models.AutoField(primary_key=True)
7      beaconName = models.CharField(max_length=10000,unique=True)
8
9      def __str__(self) -> str:
10         return str(self.beaconID)
11
12 class Category(models.Model):
13     categoryID = models.AutoField(primary_key=True)
14     categoryName = models.CharField(max_length=50)
15
16     def __str__(self):
17         return self.categoryName
18
19
20 class BeaconPosition(models.Model):
21     beaconPositionID = models.AutoField(primary_key=True)
22     beaconID = models.ForeignKey(Beacon, on_delete=models.CASCADE)
```

```
    positionDescription = models.TextField(max_length=45)
    GPSCoordinates = models.CharField(max_length=45)
    categoryID = models.ForeignKey(Category, on_delete=models.CASCADE)

    def __str__(self) -> str:
        return str(self.beaconID)

class Information(models.Model):
    inforamtionID = models.AutoField(primary_key=True)
    informationText = models.TextField(max_length=10000)
    beaconID = models.ForeignKey(Beacon, on_delete=models.CASCADE)
    datetime = models.DateTimeField(max_length=45)

    def __str__(self):
        return self.informationText
```

Listing F.1: models.py

```

from rest_framework import serializers
from .models import *

# beacon class serializer
class BeaconSerializer(serializers.ModelSerializer):
    class Meta:
        model = Beacon
        fields = "__all__"

# category class serializer
class CategorySerializer(serializers.ModelSerializer):
    class Meta:
        model = Category
        fields = "__all__"

# beacon position class serializer
class BeaconPositionSerializer(serializers.ModelSerializer):
```

```
    class Meta:
        model = BeaconPosition
        fields = "__all__"

# information class serializer
class InformationSerializer(serializers.ModelSerializer):
    class Meta:
        model = Information
        fields = "__all__"
```

Listing F.2: serializers.py

```
from django.db.models import manager
from django.shortcuts import render
from django.http import Http404
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .models import *
from .serializers import *

# Create your views here.
class BeaconList(APIView):

    def get(self, request, format=None):
        beacons = Beacon.objects.all()
        serializer = BeaconSerializer(beacons, many=True)
        return Response(serializer.data)


    def post(self, request, format=None):
        serializer = BeaconSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.
    HTTP_201_CREATED)
```

```
24          return Response(serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
25
26  # beacon details
27  class BeaconDetail(APIView):
28
29      def get_object(self, pk):
30          try:
31              beacon = Beacon.objects.get(pk=pk)
32              return beacon
33          except Beacon.DoesNotExist:
34              raise Http404
35
36      def get(self, request, pk, format=None):
37          beacon = self.get_object(pk)
38          serializer = BeaconSerializer(beacon)
39          return Response(serializer.data)
40
41      def put(self, request, pk, format=None):
42          beacon = self.get_object(pk)
43          serializer = BeaconSerializer(beacon, data=request.data)
44          if serializer.is_valid():
45              serializer.save()
46              return Response(serializer.data)
47          return Response(serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
48
49      def delete(self, request, pk, format=None):
50          beacon = self.get_object(pk)
51          beacon.delete()
52          return Response(status=status.HTTP_204_NO_CONTENT)
53
54
55
56  # cagegory list
```

```
57 class CategoryList(APIView):
58
59     def get(self, request):
60         categories = Category.objects.all()
61         serializer = CategorySerializer(categories, many=True)
62         return Response(serializer.data)
63
64     def post(self, request, format=None):
65         serializer = CategorySerializer(data=request.data)
66         if serializer.is_valid():
67             serializer.save()
68             return Response(serializer.data, status=status.
    HTTP_201_CREATED)
69         return Response(serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
70
71 # category detail view
72 class CategoryDetail(APIView):
73     # get category instance
74     def get_object(self, pk):
75         try:
76             category = Category.objects.get(pk=pk)
77             return category
78         except Beacon.DoesNotExist:
79             raise Http404
80
81     def get(self, request, pk, format=None):
82         category = self.get_object(pk)
83         serializer = CategorySerializer(category)
84         return Response(serializer.data)
85
86     def put(self, request, pk, format=None):
87         category = self.get_object(pk)
88         serializer = CategorySerializer(category, data=request.data)
89         if serializer.is_valid():
```

```
90              serializer.save()
91              return Response(serializer.data)
92          return Response(serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
93
94      def delete(self, request, pk, format=None):
95          category = self.get_object(pk)
96          category.delete()
97          return Response(status=status.HTTP_204_NO_CONTENT)
98
99
100 # api view set for beacon position
101 class BeaconPositionList(APIView):
102
103     def get(self, request, format=None):
104         beacons = BeaconPosition.objects.all()
105         serializer = BeaconPositionSerializer(beacons, many=True)
106         return Response(serializer.data)
107
108     def post(self, request, format=None):
109         beaconPosition = BeaconSerializer(data=request.data)
110         if beaconPosition.is_valid():
111             beaconPosition.save()
112             return Response(beaconPosition.data, status=status.
    HTTP_201_CREATED)
113         return Response(beaconPosition.errors, status=status.
    HTTP_400_BAD_REQUEST)
114
115 # api detail view for beacon positions
116 class BeaconPositionDetail(APIView):
117     def get_object(self, pk):
118         try:
119             position = BeaconPosition.objects.get(pk=pk)
120             return position
121         except Beacon.DoesNotExist:
```

```
122             raise Http404
123
124     def get(self, request, pk, format=None):
125         position = self.get_object(pk)
126         serializer = BeaconPositionSerializer(position)
127         return Response(serializer.data)
128
129     def put(self, request, pk, format=None):
130         position = self.get_object(pk)
131         serializer = BeaconPositionSerializer(position, data=request.
    data)
132         if serializer.is_valid():
133             serializer.save()
134             return Response(serializer.data)
135         return Response(serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
136
137     def delete(self, request, pk, format=None):
138         position = self.get_object(pk)
139         position.delete()
140         return Response(status=status.HTTP_204_NO_CONTENT)
141
142 # api set for information
143 class InformationList(APIView):
144     def get(self, request, format=None):
145         information = Information.objects.all()
146         serializer = InformationSerializer(information, many=True)
147         return Response(serializer.data)
148
149     def post(self, request, format=None):
150         serializer = InformationSerializer(data=request.data)
151         if serializer.is_valid():
152             serializer.save()
153             return Response(serializer.data, status=status.
    HTTP_201_CREATED)
```

```
154         return Response(serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
155 
156 # api view set for information details
157 class InformationDetail(APIView):
158     def get_object(self, pk):
159         try:
160             information = Information.objects.get(pk=pk)
161             return information
162         except Beacon.DoesNotExist:
163             raise Http404
164 
165     def get(self, request, pk, format=None):
166         information = self.get_object(pk)
167         serializer = InformationSerializer(information)
168         return Response(serializer.data)
169 
170     def put(self, request, pk, format=None):
171         information = self.get_object(pk)
172         serializer = InformationSerializer(information, data=request.
    data)
173         if serializer.is_valid():
174             serializer.save()
175             return Response(serializer.data)
176         return Response(serializer.errors, status=status.
    HTTP_400_BAD_REQUEST)
177 
178     def delete(self, request, pk, format=None):
179         information = self.get_object(pk)
180         information.delete()
181         return Response(status=status.HTTP_204_NO_CONTENT)
182 
183 # api view set for information details
184 class Info(APIView):
185     def get(self, request, format=None):
```

```
        try:
            bid = self.request.query_params.get('beacon')
            beacon = Beacon.objects.get(beaconName=bid)
            if not beacon:
                raise Http404

            info = Information.objects.filter(beaconID=beacon.beaconID)

            if not info:
                raise Http404

            serializer = InformationSerializer(info, many=True)

            return Response(serializer.data, status=status.HTTP_200_OK)

        except Beacon.DoesNotExist:
            raise Http404
```

Listing F.3: views.py

```

from django.db import models

from django.contrib.auth.models import AbstractBaseUser,BaseUserManager
from django.conf import settings
from django.db.models.signals import post_save
from django.dispatch import receiver
from rest_framework.authtoken.models import Token

# Create your models here.

# user:admin
#password:admin

class MyAccountManager(BaseUserManager):
```

```
16      def create_user(self,email,username,YourNumberinSchool,password=None
    ):
17          if not email:
18              raise ValueError("Users must have an email address")
19          if not username:
20              raise ValueError("Users must have an username")
21          if not YourNumberinSchool:
22              raise ValueError("Users must have an number registered in
    school")
23
24          user = self.model(
25              email=self.normalize_email(email),
26              username=username,
27              YourNumberinSchool=YourNumberinSchool,
28
29
30          )
31
32          user.set_password(password)
33          user.save(using=self._db)
34          return user
35
36      def create_superuser(self,email,username,YourNumberinSchool,password
    ):
37          user = self.create_user(
38              email=self.normalize_email(email),
39              username=username,
40              YourNumberinSchool=YourNumberinSchool,
41              password=password
42          )
43          user.is_admin=True
44          user.is_staff=True
45          user.is_superuser=True
46          user.save(using=self._db)
47          return user
```

```
48
49
50 class Account(AbstractBaseUser):
51     email       =models.EmailField(verbose_name="email",max_length=60,
   unique=True)
52     username    =models.CharField(max_length=30,unique=True)
53     phoneNumber=models.CharField(max_length=15)
54     YourNumberinSchool=models.CharField(max_length=15)
55     name=models.CharField(max_length=20)
56     date_joined=models.DateTimeField(verbose_name='date joined',
   auto_now_add=True)
57     last_login=models.DateTimeField(verbose_name='last login',auto_now=
   True)
58     is_admin = models.BooleanField(default=False)
59     is_active = models.BooleanField(default=True)
60     is_staff=models.BooleanField(default=False)
61     is_superuser=models.BooleanField(default=False)
62
63     USERNAME_FIELD='email'
64     REQUIRED_FIELDS=['username','YourNumberinSchool']
65
66
67     objects=MyAccountManager()
68
69     def __str__(self) -> str:
70         return self.email
71
72     def has_perm(self,perm,obj=None):
73         return self.is_admin
74
75     def has_module_perms(self,app_level):
76         return True
77
78
79 @receiver(post_save, sender=settings.AUTH_USER_MODEL)
```

```
def create_auth_token(sender, instance=None,created=False,**kwargs):
    if created:
        Token.objects.create(user=instance)
```

Listing F.4: (Authentication) models.py

```
from django.db import models
from django.db.models import fields
from rest_framework import serializers

from accounts.models import Account

class RegistrationSerializer(serializers.ModelSerializer):
    password2 = serializers.CharField(style={'input_type':'password'},
   write_only = True)

    class Meta:
        model = Account
        fields=['email','username','password','password2','
   YourNumberinSchool']
        extra_kwargs={
            'password':{'write_only':True}
        }

    def save(self):
        account = Account(
            email= self.validated_data['email'],
            username =self.validated_data['username'],
            YourNumberinSchool=self.validated_data['YourNumberinSchool']

        )
        password=self.validated_data['password']
        password2=self.validated_data['password2']

        if password!=password2:
```

```
            raise serializers.ValidationError({'password':'Passwords must match'})
        account.set_password(password)
        account.save()
        return account
```

Listing F.5: (Authentication) serializers.py

```

from django.shortcuts import render

from rest_framework.response import Response
from rest_framework import serializers, status
from rest_framework.decorators import api_view

from accounts.serializers import RegistrationSerializer
from rest_framework.authtoken.models import Token
from rest_framework.authtoken.views import ObtainAuthToken


# Create your views here.
@api_view(['POST',])
def registration_view(request):

    if request.method == 'POST':
        serializer = RegistrationSerializer(data=request.data)
        data = {}
        if serializer.is_valid():
            account = serializer.save()
            data['response'] = "successfully registered a new user."
            data['email'] = account.email
            data['username'] = account.username
            data['YourNumberinSchool'] = account.YourNumberinSchool
            token = Token.objects.get(user=account).key
            data['token'] = token

```

```
        else:
            data=serializer.errors
        return Response(data)

class CustomAuthToken(ObtainAuthToken):

    def post(self, request, *args, **kwargs):
        serializer = self.serializer_class(data=request.data,
                                           context={'request': request})
        serializer.is_valid(raise_exception=True)
        user = serializer.validated_data['user']
        token, created = Token.objects.get_or_create(user=user)
        return Response({'user':{

            'user_id': user.pk,
            'email': user.email,

            'username':user.username,
            'YourNumberinSchool':user.YourNumberinSchool,
        },
        'token': token.key,
        'name':user.name
        })
```

Listing F.6: (Authentication) views.py

Appendix G

Code - Others

```
package com.example.ipbeacons_mobile_app.ui.Storage;

import android.content.Context;
import android.content.SharedPreferences;

import com.example.ipbeacons_mobile_app.api.LoginResponse;
import com.example.ipbeacons_mobile_app.api.User;

import java.io.Serializable;

public class SharedPrefManager  {
    private static  String SHARED_PREF_NAME = "my_shared_preff";
    private SharedPreferences sharedPreferences;
    private SharedPreferences.Editor editor;

    private static SharedPrefManager mInstance;
    Context mCtx;

    public SharedPrefManager(Context mCtx) {
        this.mCtx = mCtx;
    }

```

```
23
24      public static synchronized SharedPrefManager getInstance(Context
    mCtx) {
25          if (mInstance == null) {
26              mInstance = new SharedPrefManager(mCtx);
27          }
28          return mInstance;
29      }
30
31
32      public void saveUser(User user) {
33
34          SharedPreferences sharedPreferences = mCtx.getSharedPreferences(
    SHARED_PREF_NAME, Context.MODE_PRIVATE);
35           editor = sharedPreferences.edit();
36          editor.putInt("user_id", user.getUser_id());
37          editor.putString("email", user.getEmail());
38          editor.putString("username", user.getUsername());
39          editor.putString("YourNumberinSchool", user.
    getYourNumberinSchool());
40          editor.putBoolean("logged",true);
41          editor.apply();
42
43      }
44
45      public boolean isLoggedIn() {
46          sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_NAME,
    Context.MODE_PRIVATE);
47          return sharedPreferences.getBoolean("logged", false);
48      }
49
50      public User getUser() {
51           sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_NAME,
     Context.MODE_PRIVATE);
52          return new User(
```

```
                sharedPreferences.getInt("user_id", -1),
                sharedPreferences.getString("email", null),
                sharedPreferences.getString("username", null),
                sharedPreferences.getString("YourNumberinSchool", null)
        );
    }

    public void clear() {
       sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_NAME,
   Context.MODE_PRIVATE);
        editor = sharedPreferences.edit();
        editor.clear();
        editor.apply();
    }

}
```

Listing G.1: SharedPrefManager.java

```

package com.example.ipbeacons_mobile_app.ui.Info_Page;

import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.ipbeacons_mobile_app.R;

public class InfopageActivity extends AppCompatActivity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.info_page);
        TextView textView = findViewById(R.id.messageText);
```

```
18
19          textView.setText(getIntent().getStringExtra("information"));
20          textView.setMovementMethod(new ScrollingMovementMethod());
21
22
23
24      }
25  }
```

Listing G.2: InfopageActivity.java

Appendix H

Code - used .xml files

```
1
2 <?xml version="1.0" encoding="utf-8"?>
3 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http
    ://schemas.android.com/apk/res/android"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:scrollbars="vertical"
8     android:orientation="vertical"
9     xmlns:app="http://schemas.android.com/apk/res-auto"
10    tools:context=".ui.Home.HomeActivity"
11    tools:ignore="ExtraText">
12
13    <View
14        android:id="@+id/wave"
15        android:layout_width="0dp"
16        android:layout_height="100dp"
17        android:background="@drawable/wave2"
18        app:layout_constraintBottom_toBottomOf="parent"
19        app:layout_constraintEnd_toEndOf="parent"
20        app:layout_constraintHorizontal_bias="0.0"
21        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/notificationText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="44dp"
        android:background="#05E61E1E"
        android:text="Your Message"
        android:textAppearance="@style/TextAppearance.AppCompat.Display1
    "
        android:textColor="#EA1414"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/wave"
        app:layout_constraintVertical_bias="0.083" />
    <TextView
        android:id="@+id/nonotification"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="28dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/notificationText" />

    <ProgressBar
        android:id="@+id/progress"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
        tools:layout_editor_absoluteX="168dp"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout_editor_absoluteY="400dp" />


    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycleview"
        android:layout_width="408dp"
        android:layout_height="474dp"
        android:layout_marginTop="56dp"
        android:scrollbars="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/notificationText"
        tools:ignore="MissingConstraints" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Listing H.1: activity_home.xml

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/imageView2"
            android:layout_width="401dp"
```

```
18              android:layout_height="200dp"
19              android:layout_marginStart="2dp"
20              android:layout_marginEnd="1dp"
21              app:layout_constraintBottom_toBottomOf="parent"
22              app:layout_constraintEnd_toEndOf="parent"
23              app:layout_constraintHorizontal_bias="0.428"
24              app:layout_constraintStart_toStartOf="parent"
25              app:layout_constraintTop_toTopOf="parent"
26              app:layout_constraintVertical_bias="0.306"
27              app:srcCompat="@drawable/logo" />
28
29
30          <LinearLayout
31              android:layout_width="match_parent"
32              android:layout_height="wrap_content"
33              android:background="@color/colorPrimary">
34
35          </LinearLayout>
36
37          <RelativeLayout
38              android:layout_width="match_parent"
39              android:layout_height="150dp"
40              android:background="@color/cardview_shadow_start_color"
41              android:padding="20dp">
42
43              <ImageView
44                  android:id="@+id/person"
45                  android:layout_width="100dp"
46                  android:layout_height="100dp"
47                  android:layout_centerVertical="true"
48                  android:src="@drawable/ic_baseline_person_24" />
49
50
51              <TextView
52                  android:id="@+id/hey"
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerVertical="true"
                android:layout_marginStart="10dp"
                android:layout_toEndOf="@+id/person"
                android:fontFamily="sans-serif-black"
                android:includeFontPadding="false"
                android:text="Hey! "
                android:textSize="20dp" />

            <TextView
                android:id="@+id/welcomeUser"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_below="@id/hey"
                android:layout_centerVertical="true"
                android:layout_marginStart="10dp"
                android:layout_toEndOf="@+id/person"
                android:text="@string/shailab"
                android:textSize="20dp" />

        </RelativeLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="347dp"
            android:orientation="vertical"
            android:padding="16dp">

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/userField"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="10dp"
                android:hint="@string/UserName">
```

```
88
89              <com.google.android.material.textfield.TextInputEditText
90                  android:id="@+id/name"
91                  android:layout_width="match_parent"
92                  android:layout_height="wrap_content"
93                  android:drawableStart="@drawable/
    ic_baseline_person_24"
94                  android:drawablePadding="10dp"
95                  android:text="shailab" />
96          </com.google.android.material.textfield.TextInputLayout>
97
98          <com.google.android.material.textfield.TextInputLayout
99              android:id="@+id/number"
100             android:layout_width="match_parent"
101             android:layout_height="wrap_content"
102             android:layout_marginBottom="10dp"
103             android:hint="@string/ipbregisterednumber">
104
105             <com.google.android.material.textfield.TextInputEditText
106                 android:id="@+id/num"
107                 android:layout_width="match_parent"
108                 android:layout_height="wrap_content"
109                 android:drawableStart="@drawable/
    ic_baseline_confirmation_number_24"
110                 android:drawablePadding="10dp"
111                 android:text="25894" />
112         </com.google.android.material.textfield.TextInputLayout>
113
114         <com.google.android.material.textfield.TextInputLayout
115             android:id="@+id/email"
116             android:layout_width="match_parent"
117             android:layout_height="wrap_content"
118             android:layout_marginBottom="10dp"
119             android:hint="@string/Email">
120
```

```
121                 <com.google.android.material.textfield.TextInputEditText
122                     android:id="@+id/mail"
123                     android:layout_width="match_parent"
124                     android:layout_height="wrap_content"
125                     android:drawableStart="@drawable/
    ic_baseline_email_24"
126                     android:drawablePadding="10dp"
127                     android:text="shailab@gmail.com" />
128             </com.google.android.material.textfield.TextInputLayout>
129 
130             <!--<Button
131                 android:id="@+id/update"
132                 android:layout_width="220dp"
133                 android:layout_height="52dp"
134                 android:layout_alignParentRight="true"
135                 android:layout_marginTop="40dp"
136                 android:text="Update"
137                 app:layout_constraintBottom_toBottomOf="parent"
138                 app:layout_constraintEnd_toEndOf="parent"
139                 app:layout_constraintHorizontal_bias="0.508"
140                 app:layout_constraintStart_toStartOf="parent"
141                 app:layout_constraintTop_toBottomOf="@+id/gotohome"
142                 tools:ignore="MissingConstraints" />-->
143 
144 
145         </LinearLayout>
146 
147         <RelativeLayout
148             android:layout_width="wrap_content"
149             android:layout_height="wrap_content">
150 
151             <Button
152                 android:id="@+id/logout"
153                 android:layout_width="110dp"
154                 android:layout_height="52dp"
```

```
155                 android:text="Logout"
156                 app:layout_constraintBottom_toBottomOf="parent"
157                 app:layout_constraintEnd_toEndOf="parent"
158                 app:layout_constraintHorizontal_bias="0.508"
159                 app:layout_constraintStart_toStartOf="parent"
160                 app:layout_constraintTop_toBottomOf="@+id/gotohome"
161                 tools:ignore="MissingConstraints" />
162
163             <Button
164                 android:id="@+id/ipbeacon"
165                 android:layout_width="110dp"
166                 android:layout_height="52dp"
167                 android:layout_alignParentRight="true"
168                 android:text="Home"
169                 app:layout_constraintBottom_toBottomOf="parent"
170                 app:layout_constraintEnd_toEndOf="parent"
171                 app:layout_constraintHorizontal_bias="0.508"
172                 app:layout_constraintStart_toStartOf="parent"
173                 app:layout_constraintTop_toBottomOf="@+id/gotohome"
174                 tools:ignore="MissingConstraints,UnknownId" />
175
176             <ProgressBar
177                 android:id="@+id/progress"
178                 style="?android:attr/progressBarStyle"
179                 android:layout_width="wrap_content"
180                 android:layout_height="wrap_content"
181                 android:visibility="gone"
182                 app:layout_constraintEnd_toEndOf="parent"
183                 app:layout_constraintStart_toStartOf="parent"
184                 app:layout_constraintBottom_toBottomOf="parent"
185                 tools:layout_editor_absoluteX="168dp"
186                 app:layout_constraintTop_toTopOf="parent"
187                 tools:layout_editor_absoluteY="400dp" />
188         </RelativeLayout>
189     </LinearLayout>
```

```
190
191
192
193 </ScrollView>
```

Listing H.2: activity_landing.xml

```
1
2 <?xml version="1.0" encoding="utf-8"?>
3 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http
    ://schemas.android.com/apk/res/android"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     xmlns:app="http://schemas.android.com/apk/res-auto">
8
9     <View
10         android:id="@+id/wave"
11         android:layout_width="0dp"
12         android:layout_height="100dp"
13         android:background="@drawable/wave2"
14         app:layout_constraintBottom_toBottomOf="parent"
15         app:layout_constraintEnd_toEndOf="parent"
16         app:layout_constraintHorizontal_bias="0.0"
17         app:layout_constraintStart_toStartOf="parent"
18         app:layout_constraintTop_toTopOf="parent"
19         app:layout_constraintVertical_bias="0.0" />
20
21
22
23     <TextView
24         android:id="@+id/messageText"
25         android:layout_width="328dp"
26         android:layout_height="523dp"
27         android:layout_marginStart="80dp"
28         android:background="#05E61E1E"
```

```
        android:textColor="@color/black"
        android:textSize="23sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/notificationText"
        app:layout_constraintVertical_bias="0.41" />

    <ImageView
        android:id="@+id/imageView3"
        android:layout_width="58dp"
        android:layout_height="67dp"
        android:layout_marginStart="16dp"
        android:layout_marginBottom="456dp"
        app:layout_constraintBottom_toBottomOf="@+id/messageText"
        app:layout_constraintLeft_toLeftOf="@+id/messageText"
        app:layout_constraintStart_toStartOf="parent"
        app:srcCompat="@drawable/ic_baseline_double_arrow_24" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Listing H.3: activity_info.xml

```

<?xml version="1.0" encoding="utf-8"?>
<com.google.android.material.card.MaterialCardView


    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="129dp"
    app:cardCornerRadius="5dp"
    android:layout_margin="8dp"
    android:background="#F4A9C0"
    app:cardElevation="3dp"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#B4EFED"
        android:padding="5dp">

        <TextView
            android:id="@+id/information"
            android:layout_width="309dp"
            android:layout_height="129dp"
            android:layout_marginStart="16dp"
            android:text="@string/
information_is_herevdlkm_dfndj_jodfndjf_djofdnkfmkfm_fkdfmndkf_dkfnd"
            android:textColor="@color/black"
            android:textSize="20sp"
            app:layout_constraintStart_toStartOf="parent"
            tools:ignore="MissingConstraints"
            tools:layout_editor_absoluteY="-3dp" />

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="308dp"
            android:layout_marginBottom="16dp"
            android:scrollbars = "vertical"
            android:textColor="#D503EC"
            android:text="@string/more"
            app:layout_constraintBottom_toBottomOf="@+id/information"
            app:layout_constraintLeft_toLeftOf="@+id/information" />


    </androidx.constraintlayout.widget.ConstraintLayout>
```

```
48 </com.google.android.material.card.MaterialCardView>
```

Listing H.4: list_info.xml

Appendix I

build.gradle file

```
plugins {
    id 'com.android.application'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.example.ipbeacons_mobile_app"
        minSdkVersion 21
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.
    AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
```

```
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }


}

dependencies {



dependencies {

    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'com.squareup.retrofit2:retrofit:2.8.1'
    implementation 'com.squareup.retrofit2:converter-gson:2.8.1'
    implementation 'com.squareup.okhttp3:logging-interceptor:4.8.1'
    testImplementation 'junit:junit:4.13.2'

    implementation 'com.android.support:animated-vector-drawable:27.1.1'
    implementation 'com.android.support:design:27.1.1'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation "androidx.constraintlayout:constraintlayout:2.1.0"
    implementation 'org.altbeacon:android-beacon-library:2+'

    implementation 'com.github.plusCubed:recycler-fast-scroll:0.3.2'
```

```
55
56 }}
```