**Mobile and Pervasive Computing  
CNT5517 - Section 1G92 & CIS4930 - Section 1376**

**Term Project Report**

**Group 1**

**Anyone There**

**Rahul Bobhate, Hiranava Das, Sharique Hussain, Hamza Karachiwala, Suryansh Singh**

**hskarachiwala@gmail.com**

**1. The Problem and its Significance**

As students, we spend a lot of our time in the library. Every assignment and exam demands a trip to this learning hub. And every trip to the library comes with its own little issues that keep troubling us at the back of our mind. Firstly, it is quite a challenge to locate a preferred spot during exam week. Some would prefer a relatively empty section to camp the night out, and the libraries at our University are massive, making it a cumbersome task to tour the entire place and identify the ideal spot. On the other hand, long hours in a quiet place to figure out complex academic challenges can be daunting. However, it may not seem as bad as it sounds if one has company. As we all have experienced in our lives as students, studying in a group is extremely beneficial – provided you have the right groups. Every time a student enters the library, we often find him or her scanning across the floor, hoping to find a known face who might be studying the same thing. Or somebody who could simply give company for a while. In the event that this does not work out, the student would definitely want others to know that he could use some company, and over time his friends could join him.

**2. The Solution Approach**

In line with the idea of a smart space, we believe that the library should be able to provide us with information. It should tell us where we could find an empty spot easily. It should be able to tell us if any of our friends are already here. It should be able to tell us if there are study groups in here which would be beneficial for us. And finally, it should be able to help us manage our study groups and remove the frustrating and time consuming dependency of making phone calls and sending text messages to do the same.

Google Beacons are great at giving us context. With a little twist, we can also use them to map somebody within a space. When a beacon provides us a context, it is also giving us a clue of our location, albeit within a range. We can use this location to identify a position within the library – in our case, a Beacon can tell us the details of the room where we would be intending to create a study group. We can then save this room information on the cloud, which our friends can then search via a mobile app. This search can be performed from anywhere; one can simply first check if any friends are in the library from home before leaving. We use a Beacon as a simple way of registering a user in the library and a means of fetching an approximate location within the space.

In order to keep track of sections in the library and the current space in them, we would need two components – sensors to track entrance and exit, and an efficient way to get that data into our system. While the physical sensor setup would be somewhat trivial, our focus is on using Google Brillo to host the sensor functionality and Google Weave to communicate with the Brillo board. We will use the Weave cloud to hold the count of people in a particular section of the library, and then export the data from there to our own cloud storage. In this way, we can exploit the Brillo-Weave ecosystem to easily bring in place an otherwise complex setup of hooking up hardware with the network and worrying about its connectivity with the cloud.

The rest of the report will outline our solution in terms of the description above. We have modelled it to solve problems in our library at the University of Florida, however, this system can be visualized as usable in other public spaces as well. For example, the problem of wanting to know about the presence of someone and their approximate location in a large park, or listing the evening activities in it can also be solved using the same approach.

**3. Design**

This section will explain the system design using the following figure -

Weave API

Service

Weave Cloud

Beacon Cloud

Mobile device

API Tool

Beacon

Nearby

Proximity

Web Service

SQL DB

Brillo

The high level system design is as follows –

1. The user will interact with our system using an Android app. This interaction includes –
   1. Checking list of friends in the library
   2. Querying their approximate location in the library
   3. Creating study groups via the app
   4. Searching for the correct group
   5. Register to an already existing study group
   6. Query for the current space availability in sections of the library
   7. Listen to nearby Beacons and receive attachment from them
2. The database on the backend of the system will hold user information (students in the case of our system) and the courses which the user has registered to. Then, it will have library information which includes the rooms that would host a study group and sections that would be common study areas. Lastly, it will have information about the study groups – the purpose of the group, its current participants and its location.
3. Interaction of the app with the database is done via RESTful APIs built using the Java Spring framework and hosted on an Azure cloud service.
4. There will be one Beacon deployed with each room in the library. This beacon will be programmed to broadcast the Room ID. The mobile app will pick these broadcasts to understand which room the user is currently in and will assist him or her with the study group creation process.
5. In order to implement a capacity map for sections in the library, ideally we would need sensors at the entrance and exit. However, we have for the time being simulated these sensors in Brillo itself by mocking a series of random entrances and exits. This simulated data is then sent to the Weave cloud. We periodically export this data from the Weave cloud to the Azure cloud service.
6. The Azure cloud service is home to our Web APIs built using the Java Spring framework. These APIs are the link between the database and the mobile app.

//screenshots in this section

**4. Implementation**

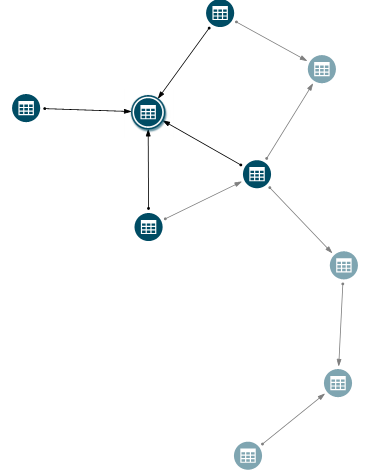
Here you describe your implementation in terms of actual work. Detailed of your implementation is to be shown here. For instance, any devices, hardware, sensors, web services, cloud services, etc. needs to be described along with their integration in sufficient details. Any algorithms used should be described here as well.

1. Google Beacon – Beeks beacons
2. Google Brillo on Intel Edison
3. Google Weave API and Cloud
4. Google Developer Console Project
   1. Proximity Beacon API
   2. Nearby API (for Beacon)
   3. Weave API
5. Android application for a User Interface
6. Azure Cloud Service
   1. Web APIs for the application
   2. SQL Server database

We start with listing out the activities associated with the setup of beacons. The first step was provisioning the beacon, where we needed to configure the beacon using an app of the manufacturer. We have set the type to EddyStone UID type, and it broadcasts this information every 1 second. After provisioning, we had to associate attachments with these beacons. We have two beacons both having one attachment associated with them. They are IDs of rooms in the library. This attachment is done in two steps, both using the Proximity Beacon API. In the first step, we register the UID of the beacon with the Google Cloud Project. In the next step, we send the data that we would want to be used as an attachment. Both these activities are done via API calls through a tool provided by Google itself. After configuring these beacons, we used the sample app provided by Google to test the correctness of the beacon setup.

For the Brillo board, we began with trying the codelabs on the Google website. After understanding the flow of Brillo, we moved on to the codelab using Weave. We thoroughly studied this flow, and identified the process with which we could simulate a sensor in the code. By mocking entry and exit data to a section, we have simulated the footfall in sections of a library. We then modified the code to add a new device representing the count in a section. Weave automatically takes care of pushing this data to the Weave cloud. Using these APIs, we have now fetched the count data from the Weave cloud and successfully stored them in our own database in the Cloud. We have taken this decision for ease of management and uniformity in the source of data for the app (App could have fetched count data from Weave cloud directly)

Moving on to the cloud service, it consists of the APIs that provide the data to the mobile applications. They are built using the Java Spring framework and are hosted in a box on Microsoft Azure. The service uses a database, again hosted in the cloud, on SQL Azure. In this way, we are adopting a completely cloud based approach and not loading our mobile clients with any data ensuring the system is lightweight. Each screen in the mobile app (discussed ahead) fetches data from the cloud using these APIs. Having such a RESTful approach ensures that we are not dependent on the type of client to use the system and can have a browser based interface also if required. The database architecture is shown in the figure –



The mobile app is targeting the Android platform. The screens are as follows –

1. Login – First time screen for a user to login into the app. It will save the user in our system as an inactive user, that is, a user who is not currently in the library.
2. View friends – This will list all the friends of the user in the library.
3. Locate a friend – This will provide a location of the room the friend is in on the floor plan of the library.
4. Create Group – It fetches the Room ID of the beacon and creates a study group associated with that room.
5. View Groups – It will list all the groups in the library currently and details associated with them.
6. Find vacancies – Will show a rough heat map of the library floor plan depicting available space; red shows a section almost full, green shows a section almost empty, and yellow shows a section filling fast.

**4.1 Implementation Effort Map**

The table summarizes our task breakup and effort taken towards each task –

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Sub Task** | **Description** | **Challenges** | **Time** |
| Beacon Integration | Provisioning | Configuring to EddyStone | Required a special app from the  manufacturer, not a very smooth  process yet and needed persistent  trying | 10 hours |
| Registration and configuration | Use the Proximity APIs to setup  beacons within the project and  configure the attachments | Novelty the only barrier | 6 hours |
| Testing with sample app | Use the Nearby API in the app  to fetch attachments | Beacon transmissions very  unreliable, hard to test | 20 hours |
| Cloud service | Try out Google App Engine | Try the offerings of the App  Engine to host a cloud service | Billing issues, plus integration issues  with Android Studio | 12 hours |
| Move to Azure | Setup Azure account |  | 2 hours |
| Build the REST APIs | Build the APIs |  |  |
| Database | Build the schema | Design the database architecture | Back and forth and some changes | 12 hours |
| Deploy to Azure | Create a working database on the  cloud | Tool setup on local machines | 10 hours |
| Mobile app | User functionality | Integrate with APIs to provide  User details and its associated  functionality | Extending on Beacon demo app  and project setup |  |
| Group functionality | Integrate with APIs to provide  Group details and its associated  functionality | Integration between team members |  |
| Search functionalities | Integrate with APIs to implement  search functionalities | Finalizing the images for the user  interface |  |
| Brillo/Weave | Setup to run demos | Hands on with the device | Flashing issues resulted in work  being stalled | 24 hours |
| Add a new state to device | Add count information to device | Limited documentation | 10 hours |
| Randomize count | Mock sensor data | Understand code flow in Brillo | 10 hours |
| Move from Weave to Azure | Moving count data from Weave  cloud to our own database for the  app | Poorly documented | 20 hours |

**5. Project Status and Challenges**

In this section, we will list the status of each module of the system –

1. **Beacons**

Completed – We have successfully completed the integration of beacons in our system. The usage of beacons as room identifiers and as a means of locating a user is implemented as visualized.

Pending – Our intention to triangulate a user in common spaces as well. This would need more beacons.

Challenge – The behavior of beacons is still unreliable and might give unexpected results.

2. **Brillo/Weave**

Completed – We have successfully included the section count data fields in our device and migrated this data from the Weave cloud to our cloud as visualized

Pending – Count data is simulated, we would need hardware to integrate actual sensors

3. **Mobile App**

Completed – The screens have been built as expected, covering all the functionalities we intended to provide.

Pending – Visual beautification and integration with a social network for Friend lists. There is scope for improvement in the flow of user interaction

Challenge – Interaction with the beacons tends to be unreliable, resulting in difficulty to design some user interactions.

**6. Demonstration**

The demonstration setup will not be too complex. We require two rooms that will have beacons in them. They will be programmed to advertise room information. One user will create a group in this room. Two users will join the group after finding it. Another user will search one of them via the app, and find the location of his friends in that group. The data for sections will be simulated continuously, and we will demonstrate how a new user can identify the best section to find a seat using this information as an example.

**7. Business Model**

As mentioned in the initial sections of the report, while this system is currently built to deploy in a UF library for student convenience, it is not the only application. Spaces need to be smarter, and need to provide us with some information about themselves, and this is a step forward in this area. The profit by having such a system is measured more in terms of human experience. Take the example of the library, students will not pay to enter the library, but having a system like this would attract more students to come to the library. This will improve academic performance, which is what any university would want. This will be the motivation for them to adopt this system. Now flip to public space like a park. An easy way to map friends in a huge park or identifying the yoga session happening in the east section of the park would be a delight for the users. And user delight is something the park authorities would be extremely interested in. The same applies to administrators of various public spaces – wedding halls, conference locations, etc. Imagine the value of having the same system for airports and waiting lounges – with groups created to while away time when flights are delayed.

The reason that we as a team are excited about this system, is that it targets public spaces which is a huge market. Public spaces are usually administered by authorities like the Government, a University, etc. who would not worry about cost when it comes to improving the experience of people within their space. This makes it a very lucrative market. The components used in our system would come at quite a low cost. Beacons, smart counters and the cloud platforms are all designed to keep costs low. Having this futuristic technology as part of the solution makes us one of the first few on the scene, having a huge market to exploit. The IoT industry is worth billions of dollars, and this solution is a component of the same booming industry. If made flexible to target all public spaces, the system can be looking at a market north of $500 million.

In order to roll out this system, we would need to iron out some flaws that would be obstacles to plug and play. For instance, we would need to use NoSQL stores for flexibility. Also, we would need a more generic user interface, possibly even one like the physical web. We would begin with running pilot projects in all the libraries at UF and possibly the neighboring universities. Based on the feedback from such spaces, we could move to more commercial spaces like amusement parks and airports which cater to more people and would affect a larger user base. Larger user bases would see more dynamic changes, that is, it would see newer users daily where economies of scale would be tested.

**8. Materialized Team Effort**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Problem/  Solution | Design / Architecture | Implementation | Reporting |
| Rahul Bobhate | 20% | 20% | 20% | 20% |
| Hiranava Das | 20% | 20% | 20% | 20% |
| Sharique Hussain | 20% | 20% | 20% | 20% |
| Hamza Karachiwala | 20% | 20% | 20% | 20% |
| Suryansh Singh | 20% | 20% | 20% | 20% |

**8. Video**

This is the link to a short demo video of this system in action at the Marston Science Library – <https://www.youtube.com/watch?v=8BuyhqfnH_c>

**9. References**

[1] Brillo documentation

[2] Weave documentation

[3] Android documentation

[4] Azure documentation

[5] Spring documentation