**Detailed design document - Fall 2017**

CS 317 - “iBeacon Tag Network Campus”

Faculty adviser: Dr. Bulut

Team members: Nicholas Bennett, Jordan Mays-Rowland, Justin Yirka

December 12, 2017

**Location of project materials:**

Source code is stored on GitHub, in VCU-CS-Capstone/senior-project-317, <https://github.com/VCU-CS-Capstone/senior-project-317>

Other files, such as status reports, meeting notes, and materials for our Sternheimer application, are stored in our team’s shared Google Drive.

Copies are stored in our Blackboard File Exchange

**Description of (agile) iterations:**

|  |  |
| --- | --- |
| September 25: | * Finalized requirements specification. * Created simple storyboards for UI. * Ordered Estimote beacons. |
| October 2: | * After research, decided to use iBeacon protocol. * Worked with Dr. Bulut to set up Apple developer account and begin learning iOS programming. * *User story - Scan for nearby BLE tags:* Implemented Bluetooth scanning on Android. |
| October 9: | * After working with Estimote SDK, concluded that Estimote is too closed-source and will not work long-term. |
| October 16: | * Prepared and submitted Proof of Concept for Sternheimer award. * Set up developer accounts with Google Firebase. * Determined that department does not have any iOS devices available for testing purposes. * Tried to program iOS using <https://www.macincloud.com/>. Concluded that it is too slow, need dedicated Mac computer. * *User story - Sign In:* Implemented Google sign-in for iOS and Android. * *User story - Record observations of Bluetooth tags* * *User story - Display past observations*: Display GPS, timestamp, and tag ID as text on screen. |
| October 23: | * Set up Mac Lab access with Dr. Bulut * *User story - Pairing*: Ability to remember specific tags. |
| October 30: | * Ordered iPhone for testing. * Set up the Firebase database. * We were notified that we were finalists for Sternheimer award. * *User story - Name tags*: Ability to label tags with custom names. * *User story - Persistent recording*: Tag observations are now stored persistently, even when app is closed and reopened. |
| November 9: | * *Backend:* Set up automated maintenance of database, truncating observations beyond a set limit, using Firebase’s Functions. * *User story - Upload* * *User story - Download* |
| November 13: | * Ordered new beacons from manufacturer in China via Alibaba. * Prepared and submitted Finalist Application for Sternheimer award. * Prepared presentation for Sternheimer award pitch. * Nov 16: In-class presentation. Practiced Sternheimer pitch. * Nov 17: Pitched for Sternheimer award committee. * Nov 20: Notified we won the Sternheimer award. Included grant for $660. |
| December 4: | * Finalized requirements for end of semester demo with Dr. Bulut. * Ordered Raspberry Pi’s. |
| December 14: | * Final demo with Dr. Bulut. * *Backend*: Set up additional database maintenace to treat heatmap information differently, deleting entries after a certain amount of time. * *User story - Heatmap*: Created a map which could be used by VCUPD to show anonymized location of all users. |

**Plans for spring semester:**

* Meet with VCUPD and determine any additional features which would be important to implement before being adopted by the school.
* *User story - security*: Implement security features as described in the paper *Techu: Open and Privacy-Preserving Crowdsourced GPS for the Masses* and which we’ve discussed with Dr. Bulut.
* *User story - account migration*: Allow users to transfer their accounts, including the cryptographic keys necessary to access their tags, between their devices.
* *User story - energy efficiency:* Experiment with various settings, such as upload frequency and scanning frequency, to reduce impact on phone battery.
* *User story - robustness*: Ensure that the app is robust to GPS imprecision and internet signal loss.
* Construct and program prototype fixed receivers using the Raspberry Pi’s we ordered.
* Thoroughly test our app, cleanup the source code and the Firebase settings, and refine the UI.
* Beta test: Using the funds allocated by the Sternheimer award, we hope to launch a beta test of approximately 100 users. This will require significant preparation, including ordering, identifying users, planning analytics, and deploying. Our aim is to demonstrate the effectiveness and usefulness of our app, as well as identify any additional bugs or future features.

**Products and technologies used:**

Estimote: <https://estimote.com/products/>

We purchased a 10-pack of “sticker” Bluetooth tags. The price was $100.

Estimote is a full-service company, aiming to sell both the hardware and software for Bluetooth applications. Unfortunately, this meant that their hardware was locked-in to their SDK; we were unable to write to the tags without their SDK. The service is free to use for the first 20 tags, but requires a contract for more. We received a quote of $1,500/year for 50 beacons, just to license the software, which made Estimote undesirable for our final product.

Bluetooth tags ordered via Alibaba: <https://www.alibaba.com/product-detail/iOS-Android-iBeacon-Smart-Anti-Lost_60440491871.html>

We purchased 6 Bluetooth tags direct from a manufacturer in China via Alibaba. The price was $66 plus $25 in shipping. Note that the shipping time was less than a week.

To order from Alibaba was different than Amazon. We had to ask Tracey to input the department’s credit card through our personal account, because Alibaba does not have one-click ordering; Alibaba uses contracts, which are negotiated over time with a particular user.

By ordering directly from the manufacturer, we avoid being locked-in to a closed SDK. However, we will now have less support in the future. The manufacturer provided us with libraries for iOS and Android to communicate with the tags.

We are still working with the tags, so no full review just yet.

iPhone <https://www.walmart.com/ip/Straight-Talk-Apple-iPhone-6-32GB-Prepaid-Smartphone-Space-Gray/329264833>

We ordered an iPhone 6 from Walmart. The cost was $200.

Originally we tried ordering from Best Buy, but the school cannot buy Apple products from Best Buy.

We ordered a “prepaid” iPhone 6, which is identical to a standard iPhone for our purposes since we will not be using a cell plan with the phone. This drastically reduced the price.

Raspberry Pi <https://www.pishop.us/product/raspberry-pi/pi-kits/raspberry-pi-zero-w-budget-pack/>

We ordered two Raspberry Pi Zero W “Budget Packs.” The cost was $30 each.

These packs were the least expensive way to order the minimum parts necessary to run the Pi. We plan to use these to build our fixed receivers/scanners.

Xcode and iOS

We have been using Xcode to program for iOS.

Note that Dr. Dahlberg suggested we use Microsoft’s Xamarin for cross-platform development. But, Dr. Bulut didn’t like Xamarin, in part because he believes it is buggy.

Dr. Bulut teaches an iOS class, and so simply added us to his Apple developer account. He also provided us with materials from the class to help get started with iOS.

The biggest trouble with iOS development has been that it requires a Mac computer.

<https://www.macincloud.com/>

This is a service which provides access to a Mac via remote desktop. We tried the free-trial, and it was too slow to merit buying a subscription.

Instead, those of us who use PC’s just visit the department’s Mac lab when needed.

Android studio and Android

We have been programming for Android using Android Studio. This has been easier, given our experience with Android development in CMSC 355.

Google Firebase: Authentication, Realtime database, Functions <https://firebase.google.com/>

We have used Google’s Firebase for our cloud based needs.

We use Firebase’s Authentication service to provide user sign-in via Google accounts, and by extension, users’ VCU email accounts. Firebase provides clear instructions for setting up Authentication as well as easy to use libraries.

We use Firebase’s Realtime database to store observations of users’ tags. The database structure is simple JSON, and it is very easy to set up. Again, clear instructions and code samples are provided for both iOS and Android. We note that by using the Firebase library, we can even avoid the details of local storage, as the library can create and manage a local database on a device. This service is free up to a particular amount of storage and bandwidth. We still need to do testing, but we think it’s possible we may be able to run our beta test without a subscription.

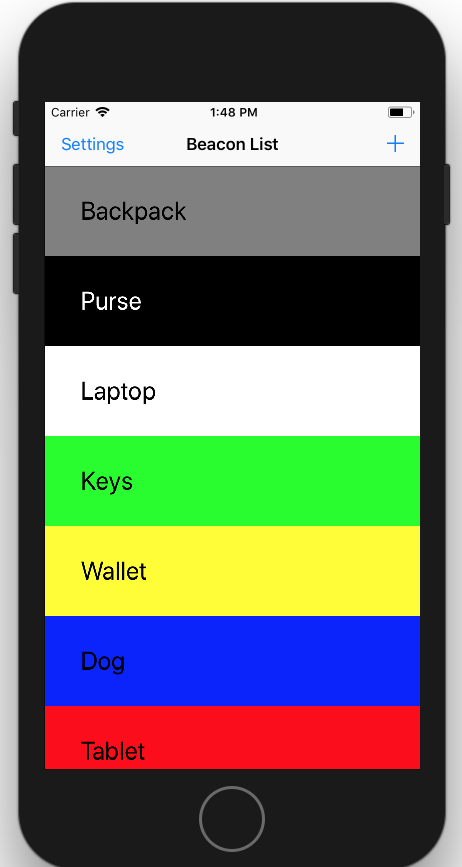
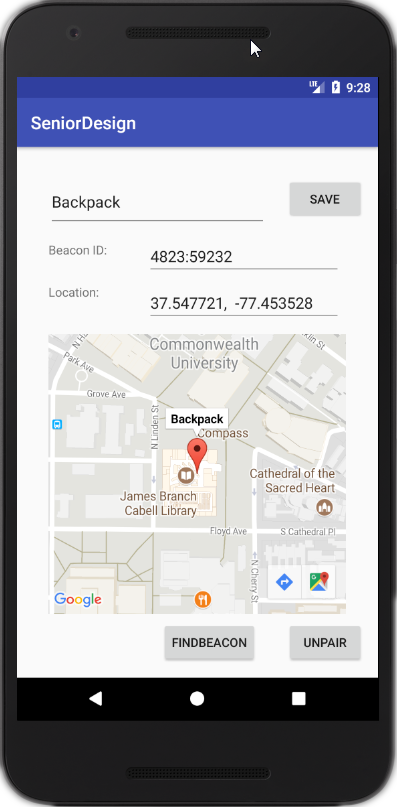
We use Firebase’s Functions to perform automated maintenance of our database. Specifically, we automatically truncate old records. This service also is free up to a particular number of function calls per day. We are more concerned about the quota for this service, and will either require a Firebase subscription for our beta test or will need to find a way to reduce the number of function calls.

**Architecture:**

Example UI:

The first picture displays a list of tags that the user owns. The user has named each tag by whatever item they are tracking.

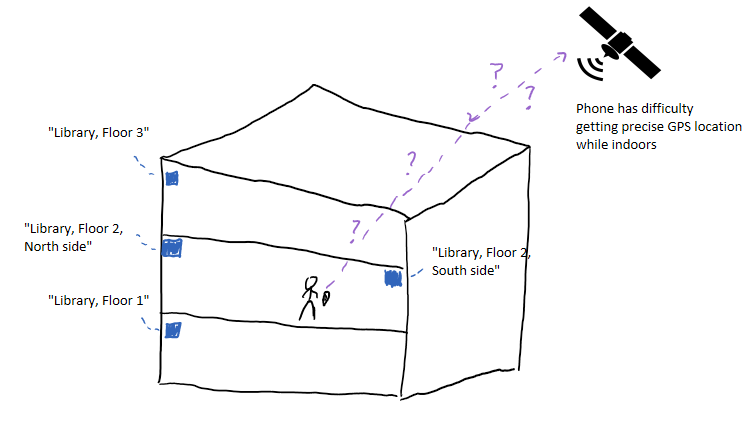
By selecting a partiuclar tag, the user is brought to the screen in the second picture, displaying the last known location of the tag. This location may be reported by the user’s own phone, or by any other user on the network.

Fixed receivers/scanners design:

The concept behind the fixed receivers is simple. Users’ phones tend to have poor GPS precision when indoors. Further, GPS is rarely able to determine elevation to the accuracy of a particular floor in a building.

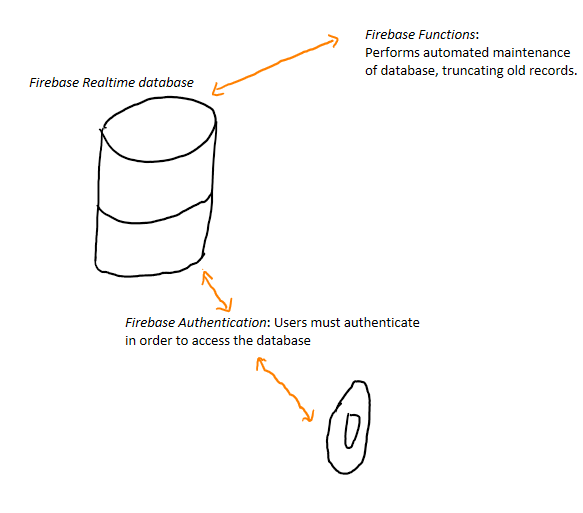
Because our system is customized to a particular campus, we suggest installing units, possibly Raspberry Pis, which are aware of their exact location. These will scan the area for nearby Bluetooth tags, reporting just like any user might, but with the addition of a string such as “Engineering, Room 301.”

****

Database design:

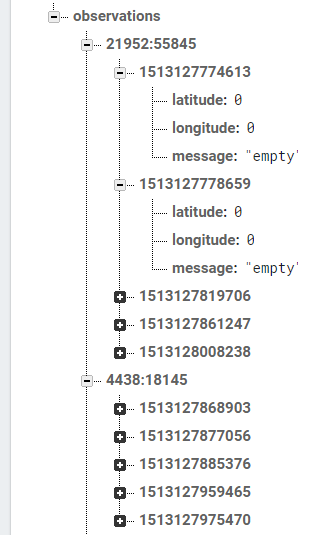
The cloud features of our system involve three services offered by Google Firebase.

They include Authentication, Realtime database, and Functions. Their relationships are described below.



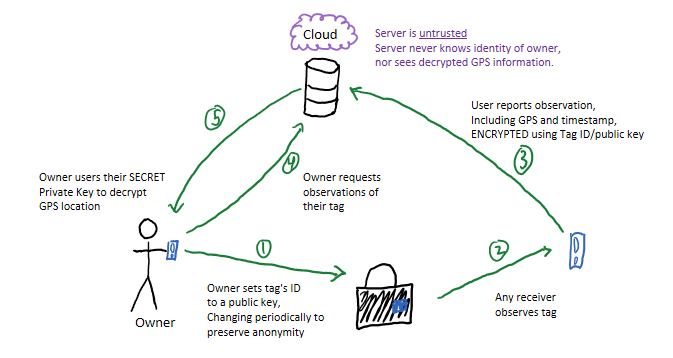
The database’s structure is defined by JSON, and so is a tree. The structure is

{tag ID} - {Unix timestamp of observation} - {latitude, longitude, message (fixed receiver)}.

****

Security design:

We plan to implement security considerations as described in the paper *Techu: Open and Privacy-Preserving Crowdsourced GPS for the Masses.* The high-level design is below.

****