UNIVERSITY OF CALGARY

ENGO 500: GIS and Land Tenure #2

Technical Deliverables Report

By: Jeremy Steward
Kathleen Ang
Ben Trodd
Harshini Nanduri
Alexandra Cummins
Supervisor: Dr. Steve Liang

DEPARTMENT OF GEOMATICS ENGINEERING

SCHULICH SCHOOL OF ENGINEERING

12/06/2013

Table of Contents

List of Figures

Index of Tables

Introduction

Purpose & Scope

The purpose of this document is to summarize and describe the progress of group *GIS & Land Tenure #2*, in particular to report the development of an open-source Internet-of-Things initiative. This report describes in detail the work done by the group with regards to the planning process of the project, the development of a prototype smart-shelf as part of the project design, and the development of a user-friendly web-based interface with which to interact with our shelf prototype.

Upon completion of describing the work done thus far in the project lifetime, the project schedule and timeline is evaluated. Milestones accomplished and problems encountered are discussed and evaluated, and finally a risk-assessment for the future of the project lifespan is considered. A final summary of the project status is stated, followed by a brief conclusion on the work done and goals to strive for in the remaining life of the project.

Project Overview

The particular focus for this project is about the concept of “Internet of Things”. The Internet of Things (IoT) is a scenario in which objects, animals or people are provided with unique identifiers and the ability to transfer the data automatically over a network without requiring human-to-human or human-to-computer interaction [1]. The Internet of Things, while barely known within the common vernacular, is an interesting concept in that it suggests a world where information about the real world can be directly interacted upon through common interfaces (in this case, the internet).

The number of IoT applications are expected to rapidly increase in the future, and likewise drive tremendous value for both businesses and people [2]. However, a budding problem within the IoT ecosystem is that for any given object connected to the IoT, we find that there often exists implementation specific methods. Similar to how HTTP unified the web, part of this project focuses on implementing an Internet of Things application in a standards-compliant fashion, to reduce the amount of intellectual overhead for developers who wish to learn how to interact and develop the IoT platform, as well as provide users a common gateway for IoT devices that they can trust.

To this end, our primary objective for this project is to develop an application that applies Internet of Things concepts in order to serve as an example for future Geomatics professionals and developers to create location-aware services, and likewise to assist in pushing forward the IoT SWG standard [3] proposed by the Open Geospatial Consortium (OGC). OGC standards are technical documents that specify interfaces and/or encodings. Software developers use these documents in order to build open, accessible interfaces and encodings into their products and services [2]. Given these objectives for our project, two explicit requirements were set:

1. Develop a hardware prototype of some sensor setup that uses the Internet of things, and interface it with some web-based software system using the OGC IoT SWG standard [x].
2. Develop this application as Free, Open-Source Software (FOSS), such that anybody can access the code and tweak, develop, mashup, and redistribute it past the lifespan of the project.

From these, our group was given freedom to plan and manage what kind of project we wished to develop.

Project Conceptualization and Planning

Given the above stated goals of our project, our team was given free-reign over what type of solution we believed would be a good fit to act as both a proof-of-concept and working implementation of an Internet of Things application. It was important to consider that our project would not only act as an example case for the upcoming OGC IoT standard, but we also wished to design something that would have impact, and hopefully be useful in some aspect.

That said, much time was devoted to determining an appropriate application with which to apply IoT principles to produce a solution. For the majority of the first six to eight weeks of development, serious consideration went into specifically defining and refining our project scope and definition. Most of this progress has been consolidated into an appropriate literature review [x], which likewise defined both our prototype and web-interface specifications, as well as proved to show that our project would have impact and provide value to store clerks and managers who would be the primary target of our developed solution [4] [5].

Based on our scope and purpose, our group divided the project into two major categories: Our shelf prototype development, and the development of the web-based interface (website). Overall, these two facets of our project are connected through the use of the OGC IoT SensorThings API as seen in Figure 1:

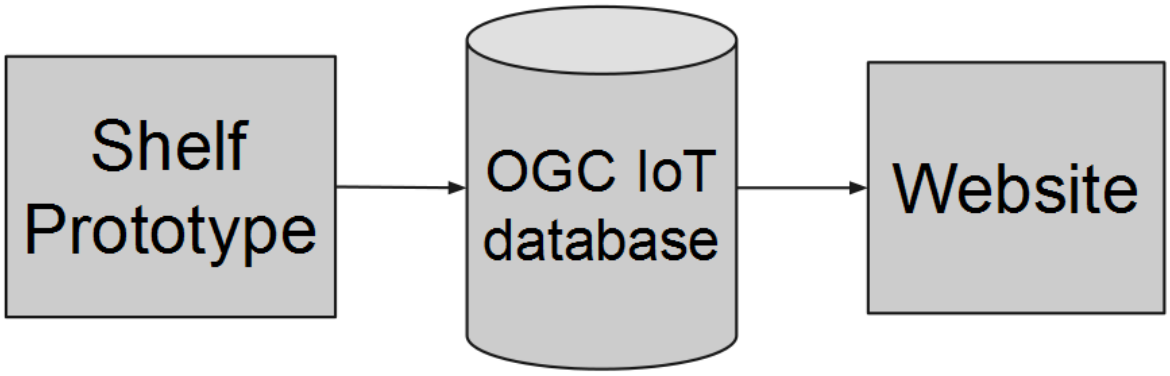

Figure 1: Division of project priorities

For each of these project components, a use case analysis was conducted. A use case study is done to define interactions between a system and an actor or actors to achieve a goal. This is a high level-planning tool to help develop requirements. It was chosen as a first step in development due to its precedence in modern software engineering as well as its ability to evolve with the project.

The first step taken was to create a use-case or class-interaction diagram. Creating a visual overview of the system helped to define what functionality the final system would support. Figure 2 shows this Use-Case diagram for the Shelf Prototype:

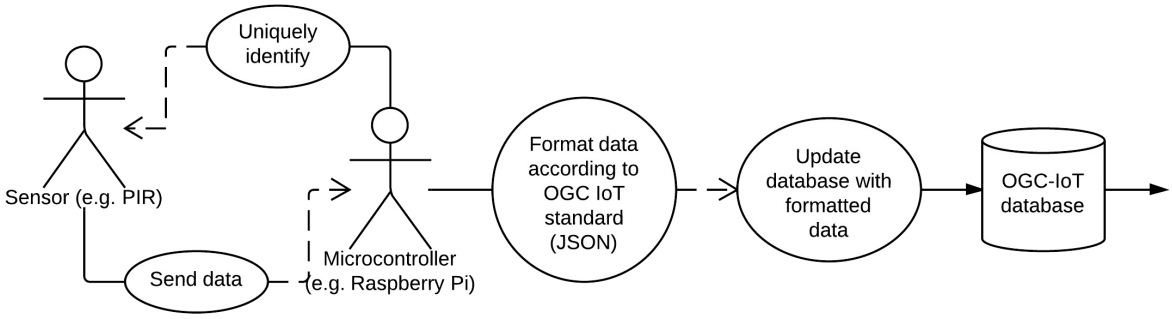

Figure 2: Use-case diagram for sensor prototype

The corresponding use-case diagram for our website can be seen in Figure 3 below:

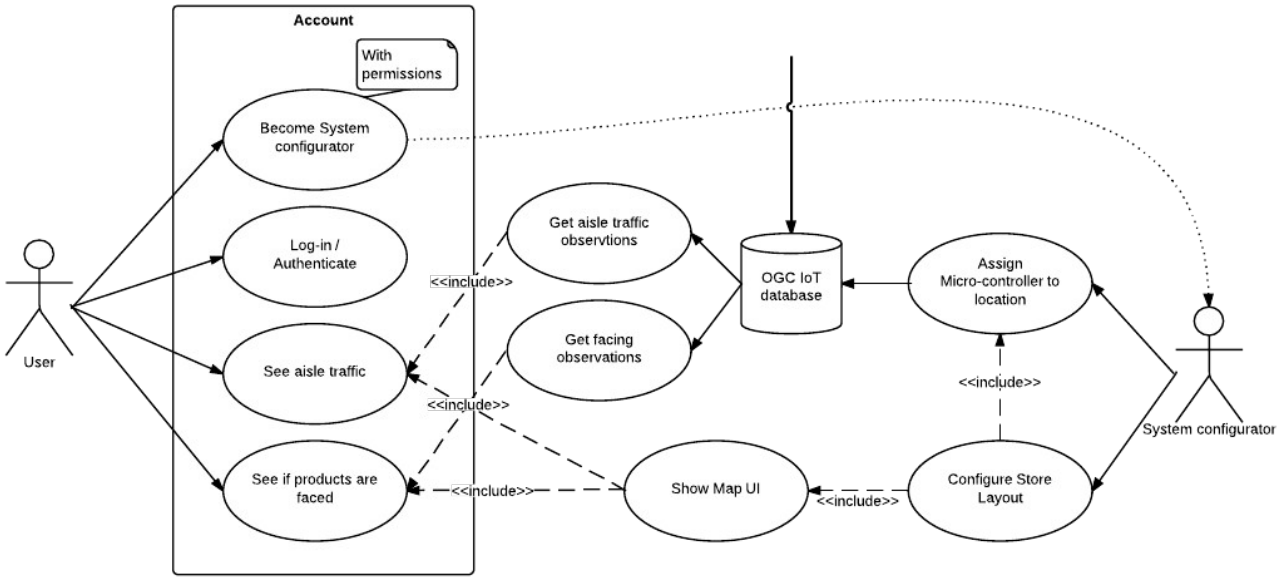

Figure 3: Use-case diagram for the website

In the diagram, the actors are represented by the “stick-man” figures, whereas the use cases are depicted by the ellipsoidal elements. The other elements, such as the database, exist to show the link between the hardware and software use cases. The “Account” entity, and “With permissions” note are shown for clarity purposes.

Having completed the diagram, the team worked at preparing a use case for each of the identified actions that the system will be responsible for. See Tables 1 and 2 below:

Table 1: Shelf prototype use cases

1.1	Update database with formatted data
1.2	Format data according to OGC IoT standard
1.3	Uniquely identify sensors
1.4	Send data to micro-controller

Table 2: Website use cases

2.1	Assign location to a micro-controller
2.2	Configure a store layout
2.3	Get aisle traffic observations
2.4	Get facing observations
2.5	Get aisle traffic
2.6	See if products are faced
2.7	Show map UI
2.8	Log In/Authenticate
2.9	Become System Configurator

The majority of these use cases can be seen in detail in Appendix A and B. While the initial phase of use case analysis has been completed, the vision for the system is not set in stone. As noted previously, these are subject to change with the evolution of the project.

Having defined use cases opens up the possibility of beginning on UML case diagrams. This will be the next step in project planning to be tackled by the team. So far, only initial research into the requirements of making a UML diagram has been completed. For each use case, classes will be created to define the attributes and methods/operations needed to achieve the goals of the use case. The UML Class diagram will be created in UML 2.2 style as defined by the Object Modeling Group (OMG) [6] [7].

Location-Aware Shelf System (LASS) Prototype

In order to advance the hardware side of the design, an official equipment list has been prepared after much planning and budgeting of costs. Previously, the team had obtained sufficient equipment with which to experiment, including a micro-controller (Raspberry Pi) and several temperature sensors. Although the team has conducted some testing with the sensors available, our primary concern is to get hold of a few PIR Motion Sensors and Magnetometers, with the help of Dr. Steve Liang, in order to start collecting required data. A PIR (Passive Infrared) motion sensor is an electronic sensor which measures infrared light radiating from objects in its field of view and it is used to sense movements of people, animals, or other objects. An example of a PIR Motion Sensor can be seen in Figure 4 below [8].

Figure 4: Example of Passive Infrared Motion Sensor

A magnetometer is a measuring instrument that is used to measure the strength, and the direction of magnetic fields. A magnetometer can be used to check the shelf facing as they can detect magnetic metals. For the purposes of our project, we aim to mount these magnetometers onto self facing shelf racks, which will provide specific magnetometer readings based on whether or not the shelf is empty. An example of a magnetometer can be seen in Figure 5 [9]:

Figure 5: Example of Magnetometer Sensor

In addition to hardware specifications, the team was also given access to Dr. Steve Liang's database on GitHub (OGC IoT GitHub) [3]. Time has been investing in learning and practicing with Python programming, and a skeleton code has been written, which will enable the creation of 'things' and update them to the database. From this we were able to observe the relationships of entities of each 'thing' that has been created. This understanding is vital to our completion of the skeleton code and to the transfer of our collected data to the website.

Specifications

The following two lists of functional and non-functional specifications were compiled in our previous literature review [10] for the Hardware section. These specifications seek to define the direction with which our final product will be developed.

Functional

1. The proposed shelf should be able to tell when a product is no longer faced. For example, in the case of spring load racks, magnetometers could be attached to the back of the spring loaded wall of the rack, and magnets at the face of the shelf could be fixed such that the magnetometers reach specific readings when the shelf is unfaced.
2. The proposed shelf should be able to determine if somebody is standing in-between the length-wise area of the shelves, such that it is possible to know whether a customer is standing between the two edges of the shelf-space. This makes it possible to know where in an aisle a customer might be. An example might be to use range or motion sensors at each edge of the shelf. If a customer crosses one side but not the other, this could indicate that they are in between both sensors.
3. The shelf should be sized similarly to existing store shelves, and should not provide any lower limits on the amount of stock that can be placed on a shelf. In this same way, the shelf should not obstruct or otherwise place limits on the amount of space between aisles compared to that of existing shelves.
4. The final device should not break without internet access, but should expect reliable internet infrastructure in place in order to perform.

Non-Functional

1. The final product should conform to the Open Geospatial Consortium standard for IoT SWG. Specifically, the data output from the hardware should be interoperable with a conformant database built from this standard.
2. Relatively stable internet access should be available (either through a wired or wireless connection) so that the shelf can update its status as frequently as possible.
3. Power to the shelf should be made available. For a single shelf consisting of one Raspberry Pi controller, a 700 mA (5V) power source is necessary to operate.

Web-Based Interface for LASS

The key word that many take away when imagining the Internet of Things is the *internet.* The internet has exploded in use and popularity in recent years, and plays an important part in information exchange. Therefore, as a corollary to our LASS prototype which takes measurements in from attached sensors, we are likewise developing a web-based user interface with which to interact with our shelf systems. This interface will exist as a website where targeted users can log in, view, and interact with the hardware through their web browser.

Specifications

The following two lists of functional and non-functional specifications were compiled in our previous literature review [10] for our website design. These specifications seek to define the direction with which our final product will be developed.

Functional

1. The ability to view shelf facings in real time, and determine where products in the store need to be restocked.
2. The ability to track customers entering and exiting each respective aisle, and determine how long each customer stayed in that section of the store.
3. The ability to generate some basic analytics based on data from the sensors. This can include items such as determining hotspots or busy times around the store, analyzing which parts of the store are more popular (most shelves empty), and other organizational information, such as busy times within the store, or average time left un-faced.

Non-Functional

1. The system should be painless and simple to introduce. Specifically, it should not require the user to do anything beyond setting up the hardware. For this reason, a central website that distributes management of multiple entities is preferred to a more decentralized system, where entities would set up their own specialized hardware for their application.
2. The system should be secure and prevent unauthorized entry. This is to protect consumers from privacy concerns, as well as to protect both managers and the data server from malicious entities.
3. The system should be robust enough that events such as power loss or switching products between shelves should not produce any significant overhead to the system.
4. The system should be easy to manage and should provide managers with the ability to readily pull up any information from a given store or branch that they might choose.
5. The system should have received an extensive amount of testing, preferably upwards of 30 hours of actual use testing and bug-fixing. As the project expands the test requirements will also be expanded to maximize ease-of-use of the final product.
6. All of the software should operate using the OGC IoT SWG standard, and should be published as free, open-source software for anyone to access and improve upon.

Completed Work-to-Date

Aside from the above listed specifications, and the aforementioned use-cases describing the interactions of the system as a whole, other work has been started in the interest of rapidly developing the web-based interface of our project. In particular, research regarding web technologies and frameworks have been progressing far ahead of schedule, thanks in part to splitting the team and using a divide-and-conquer approach for the remainder of the project milestones.

With regards to research, the website team has decided to develop the final system using the following popular, established web technologies:

- HTML + CSS + Javascript

- Bootstrap [11] for dynamic CSS
- jQuery [12] for dynamic animations and effects in our user interface (UI)
- D3JS [13] for visualization of our sensors

Of the above, the website team has already learned and become comfortable with HTML, CSS, and Javascript as development languages, and jQuery as a standard development library. Thus, much of the research for implementing the website is done, all that remains is to learn the remaining two technologies (Bootstrap and D3JS) and implement our use cases as defined in Appendix B.

Technical Demo of OGC Database Interaction

To show off the progress made in understanding the OGC IoT SWG database, and to show off an example of how measurements are to be read off of the server from live data, a small technical demo page was created with a small number of interactive elements to show live data being read from the server in real-time. An example of this page can be seen in Figure 6 below:

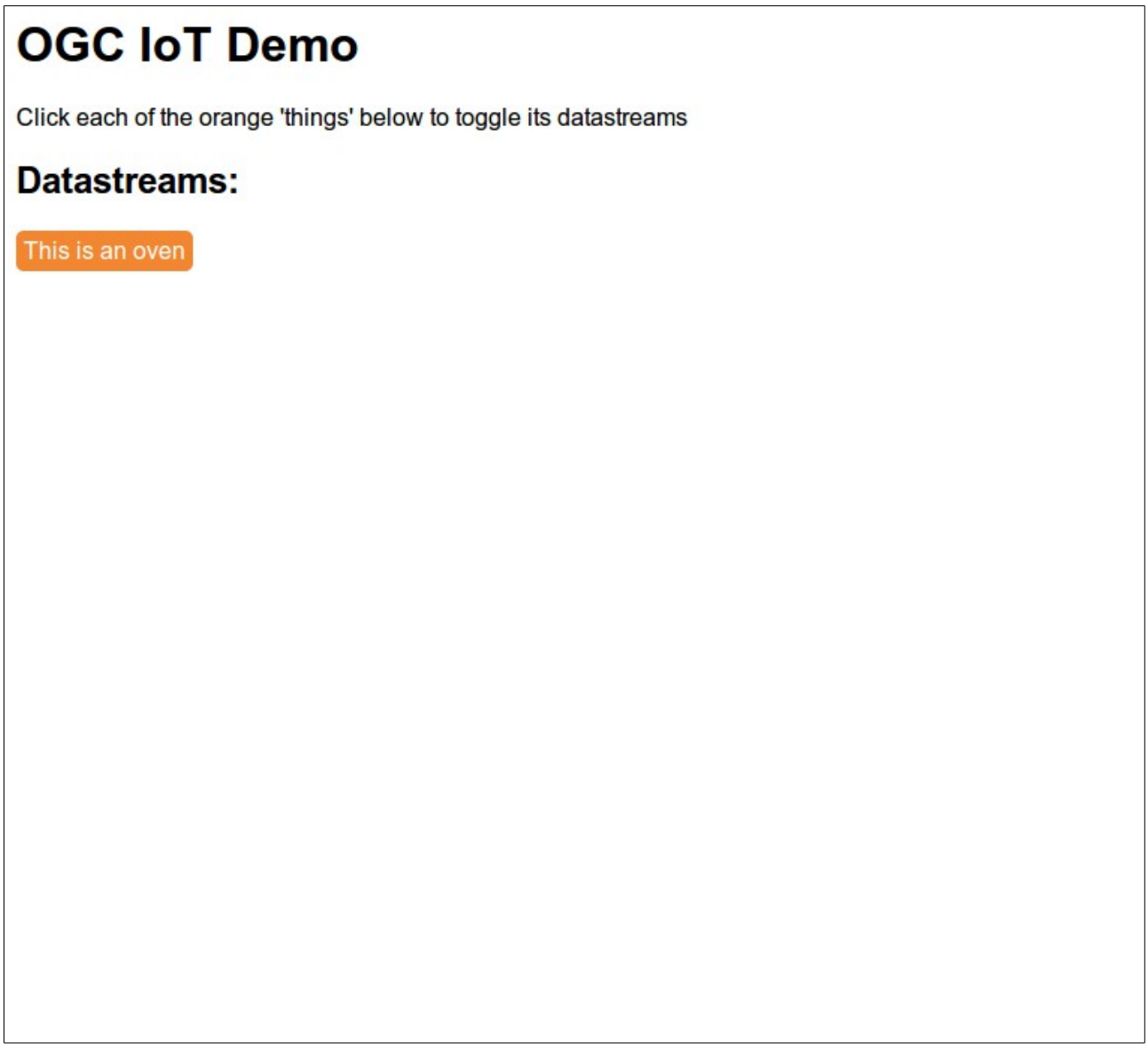

Figure 6:An example of the web page rendered before user interaction

From the above webpage, clicking on the orange button shown above yields the following data-

streams pulled from the provided server, seen in Figure 7:

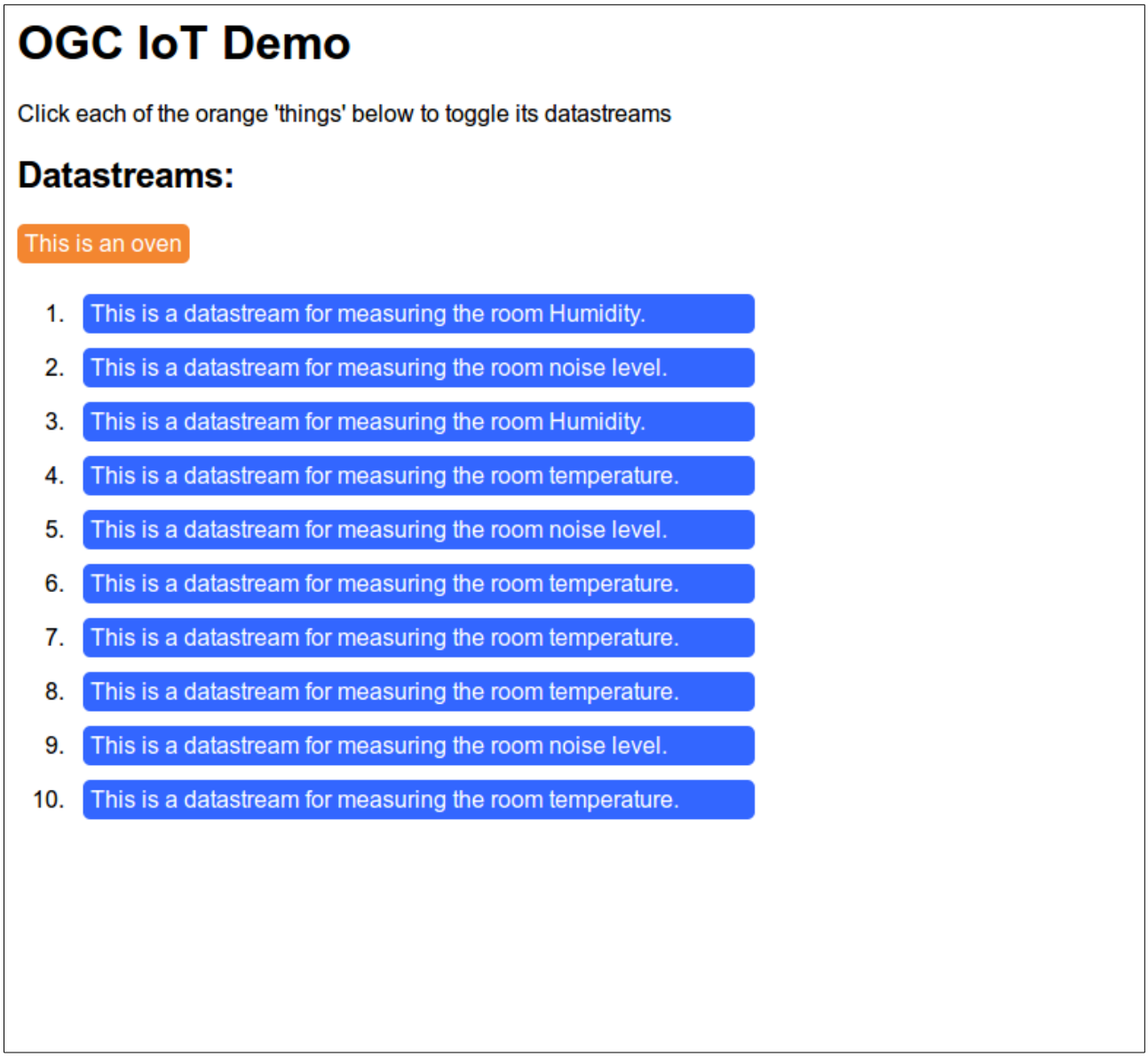

Figure 7: Dynamic loading of observation content (descriptions of data-streams shown)

While not directly applicative to the end goals of our interface, this example shows our progress in pulling data directly from the server, which is incidentally one of the larger hurdles in accomplishing our end goal. The full code can be found on the group Github repository [14], or can be seen at http://jsfiddle.net/XkBAm/1/.

Schedules, Risks and Status

The project has gone through a significant change from the original forecasted schedule of events. This was primarily due to a lack of experience in design, development procedures, and project management within the team. Although the tasks completed did not align well with the original timeline found in the project proposal, the team remains confident that the goals of the project will be met by the final deadline.

The biggest mistake in the projected project velocity was in identifying and estimating the initial stages of the project. With a loosely defined project statement, the team spent a considerable

amount of time deciding on an application for which an IoT solution was appropriate. The tasks thats followed, such as deciding on which micro-controller to use, which sensors to use, what website framework to work with and learning the OGC's proposed IoT API were all time consuming tasks that had originally appeared to be small amounts of work in terms of technical deliverables. However, they did result in a greater knowledge and understanding of each of the components that make up the project, as outlined in the preceding section. These were not all anticipated steps, and thus, are not shown in the original Gantt Chart.

Although the team had planned on doing the project in two phases, hardware development followed by website development, this plan was deviated from. At week 8, the team decided to split into a team of three working on hardware implementation, while a team of two began work on the product's website. This made a considerable change to the timeline of the project.

Examining the Gantt Chart form the project proposal shows that the following milestones in Table 3 below were projected to be done by December 6th:

Table 3: Milestones to be completed before end of term (December 6th)

Milestone	Status	Comments
Determine location based application	Complete	The application chosen was Location-Aware Shelf System (LASS)
Design sensor setup - Determine technical requirements - Acquire Sensors - Create prototype - Test prototype	 Complete In Progress Not started Not started	The technical requirements were determined as part of the deliverables for the literature review. The required sensors have been sent for approval but not acquired.
API-Sensor interface - Research into languages/specifications - Collect test set of data	 Complete Not Started	While the language and specifications have been researched, having not purchased sensors leaves this task unfinished.
Project Proposal	Complete	
Literature Review	Complete	
Technical Deliverables - Written Report - Oral presentation	 Complete Complete	

The following milestones in Table 4 were projected to have been started by December 6th:

Table 4: Milestones to have been started by end of term (December 6th)

Milestone	Status	Comments
Create Library to interface with database	In Progress	Using simulated data, information has successfully been PUT to the database

Table 5 below depicts milestones that were projected to begin *after* December 6th, but have been begun development due the introduction of a split team focus:

Table 5: Milestones projected to being at the start of the second term, but have already been started

Milestone	Status	Comments
UI Development - Research required frameworks - Create prototype of UI - Attach to database - Test UI - Polish / Revise UI - Create video of UI in use	 Complete Not Started In Progress Not Started Not Started Not Started	Progress to date focused on learning website development, and interfacing with the database. A mock website was created, but UI development has not yet begun.

Lastly, we can see Table below, which describes milestones and sub-tasks that were not initially predicted, but were completed before December 6th:

Table 6: Milestones not initially accounted for but have been completed or are in progress before the end of term (December 6th)

Milestone	Status	Comments
Define Use Cases - Hardware Use Case Diagram - Hardware Use Cases - Website Use Case Diagram - Website Use Cases	 In Progress In Progress In Progress In Progress	While the Use Cases attached in Appendix A and B at the end of the report are mostly complete, they have yet to be finalized by our supervisor, and some additional ones remain to be added at this moment in time.
Learn web development fundamentals - HTML - CSS - jQuery	 Complete Complete Complete	This milestone was for the two team members focused on web development.
Learn introductory Python programming	Complete	This milestone was for the three team members focused on hardware development
Explore and understand database structure and commands	Complete	
Create demonstrative proof of concept - Post data to server - Display data on website	 Complete Complete	This milestone was done for the oral presentation

Due to the large change in the project direction, a new Gantt chart has been created. The time that has passed now reflects the project milestones that have been accomplished thus far, and a re-evaluated version of the future milestones can now be seen past week 12. This Gantt chart can be seen in Figure 8, on the following page:

Figure 8: Updated Gantt chart of project timeline. Current time is marked by the solid gray bar, tasks that have been set back are marked in maroon.

Looking at the original Gantt chart seen in the proposal [15], this one has several differences, most notably in that it is far less vague and is more specific with the project requirements and scope, which have been more properly been defined since that time. The major differences are:

- The original *Application Design* section was expanded to the new *Project Conceptualization & Planning* section, which lays out several sub-tasks within the milestone, including:
  - Research into the value of chosen application
  - Research applicable sensors
  - Produce specification sheet(s)
  - Develop and flesh-out our class (UML) model diagrams.
  - Build use-cases to define the structure and interactions between our software elements
- The *Design sensor setup* sub-task has been split from the planning phase of the project, and more appropriately placed into the *Application Prototype Development* section. The sub-tasks of this milestone have likewise been merged with the *API-sensor interface* milestone, since a large portion of the work done in creating our prototype will revolve around ensuring that our prototype can properly communicate with the provided OGC IoT database.
- The website development has been consolidated, and proper testing steps (including a usability survey) have been added as sub-tasks.

Potential Risks

Some of the potential challenges we risk running into can be summarized in the following list:

1. Given that we have delayed ordering sensors until really late, we potentially run the risk of losing much of our time waiting for the sensors to ship. Our application prototype currently allots for some time to be lost (see Figure 8), however, there is only approximately a three week window where this is acceptable. Should the sensors be delayed by more than three weeks (from their regular shipping time, 4-6 weeks), there may be serious consequences to the final leg of our development. To mitigate this, we are using the inputs and outputs specified by our use-cases to simulate data read in from the sensors we currently do not have. In this way, we should still be able to make a functioning system, which should allow other pieces of the system to interact even if the physical sensors themselves are missing.
2. Since our group has split into two teams, one focused on developing the prototype, and the other focused on developing the web-based interface, we run the risk of developing orthogonal to each other's work. In this case, we require strict definition and adherence to our use cases in order to avoid running into a situation where the input of one of our components is not coherent with the remainder of our system. In this way, by mitigating this risk, we also save ourselves from code duplication, and prevent ourselves from running in circles re-writing modules and objects.
3. One significant challenge during this term was gaining some momentum within the project, specifically because much of the legwork this term favoured planning, careful research, and conceptualization over hard deliverables. This resulted in appearing to have very little done on the surface, while having established a reasonable framework through our use-cases and documentation with which to

develop our project further. Given that there are a larger number of breaks in the winter term (Christmas break, Reading week, Easter, etc...), we run the chance of losing our momentum and potentially falling behind. What's more, with this being the last term before (hopefully) graduation for the group, there is a chance that work from prioritizing other classes and electives constitutes a significant barrier to getting work done fluidly and efficiently. For this reason, a more stringent time input will likely be enforced in the new term, and expectations for weekly output will attempt to be measured to some standard. Moreover, much of the work into the finalization and development of our use-case modules will attempt to be done over these breaks, to try and mitigate the otherwise demanding nature of engineering.

4. One risk that was not fully explored in this document or in others we have published regard that of cost. It is assumed until this point that our prototype will be built, but little thought has been given to cost aside from ordering the sensors. This is a significant risk in that if we want our final product to be successful and have impact, we need to worry about the cost of the final device and the associated ease-of-use and value it provides. The value itself has been explored within our previous Literature Review [10], but has not necessarily been weighed against the cost of individual sensors (or that of mass-produced products, which would likely bring about better results). Part of the final report should perhaps explore what was learned with regards to the number of sensors required for effective testing, and what the lower limits are in terms of cost reduction.

Conclusion

Overall, the evolution of the project, as well as the management principles of the project have shifted greatly from the original intent planned back during the project proposal. Accomplished milestones within our planning phase, prototype development, and web-based interface (website) development were explored, and progress was evaluated. Some specific roadblocks were encountered that hindered our expected progress in the planning and prototype development milestones; specifically, use-case diagrams became necessary to mitigate further risk, and a late determination and order time of sensors has the potential to set back the progress of the prototype development. However, despite these challenges, by reorganizing the group into two teams to tackle the remaining tasks, significant headway was gained in the website development phase, particularly with regards to the research required to learn web technologies and frameworks.

Despite the progress made, some things could have been handled differently, most notably the amount of contact kept with our supervisor (Dr. Steve Liang) throughout the semester. It was difficult attempting to get a hold of Dr. Liang, in particular due to his travel and work schedule. We were often limited to bi-weekly meetings, which in the future may be more beneficial as weekly meetings. The amount of software training also proved challenging, as many of the group members were not acclimated to Github so easily. In retrospect, more time could have been spent properly training each group member to ensure that each member understood how to use and manipulate files and revisions through using git and Github.

Appendix A: Shelf Prototype Use Cases

Use Case ID:	1.1
Use Case Name:	Update database with formatted data
Created by:	AC
Date Created:	24-Nov-13
Date Last Updated:	
Last Updated by:	
Actor:	Raspberry Pi and sensors as they store and collect data
Description:	Once the hardware is set up and working, data will be collected by the actors and stored. As this happens it should be formatted if need be (use case 1.2). The database should then be accessed and updated with new, formatted data. This will eventually be implemented in real time.
Preconditions:	1. Sensor is connected and collecting data
	2. Database is accessible
	3. Data has been successfully formatted
Postconditions:	1. Database received the new values successfully
Priority:	High during real time updates, otherwise normal
Frequency of Use:	High
Normal Course of Events:	1. Data is collected and stored by actors
	2. Data is rendered into a database compatible form
	3. Database is accessed and data is transferred
Alternative Courses:	1.a Data is collected but not stored
	1.a.1 User is notified of the error
	3.a Data is transferred but is not formatted correctly
	3.a.1 User is notified
Exceptions:	
Includes:	1.2 Format Data
Special Requirements:	
Assumptions:	1. Assumes that the collected data needs to be formatted. Will be verified through testing.
	2. Assumes that the database (can be accessed and) does not clear itself every 10 minutes.
Notes and Issues:	Much of this may change as the group begins working with sensors.

Use Case ID:	1.2
Use Case Name:	Format data according to OGC IoT standard
Created by:	AC
Date Created:	25-Nov-13
Date Last Updated:	
Last Updated by:	
Actor:	Raspberry Pi and sensors as they store and collect data
Description:	Data collected by sensors will be formatted so that it can be easily updated to the database with use case 1.1
Preconditions:	1. Sensor is connected
	2. New data is being gathered and stored
Postconditions:	1. Database received the new values successfully
Priority:	High during real time updates, otherwise normal
Frequency of Use:	High
Normal Course of Events:	1. Data is collected and stored by actors
	2. Data is rendered into a database compatible form
Alternative Courses:	1.a Data is collected but not stored
	1.a.1 User is notified of the error
Exceptions:	
Includes:	1.1 Update database
Special Requirements:	
Assumptions:	
Notes and Issues:	Much of this may change as the group continues working with sensors.

Use Case ID:	1.3
Use Case Name:	Identify sensors
Created by:	AC
Date Created:	25-Nov-13
Date Last Updated:	
Last Updated by:	
Actor:	Raspberry Pi and sensors attached to it
Description:	All the sensors collecting data need to be recognized and given a unique ID so that their data may be distinguished even after it is formatted and updated to the database
Preconditions:	1. Sensor is connected and recognized
Postconditions:	1. The data that was collected can be related to a unique ID that represents each sensor
Priority:	High during real time updates, otherwise normal
Frequency of Use:	High
Normal Course of Events:	1. Sensors are connected and recognized
	2. Sensors are assigned a unique ID
	3. Data is collected and the ID of the sensors corresponds to it
	4. All this is relayed to the formatting use case (1.2)
Alternative Courses:	1.a Data is collected but not stored
	1.a.1 User is notified of the error
Exceptions:	
Includes:	1.2 Formatting Data
Special Requirements:	
Assumptions:	
Notes and Issues:	Much of this may change as the group continues working with sensors.

Use Case ID:	1.4
Use Case Name:	Send data to micro-controller
Created by:	BT
Date Created:	5-Dec-13
Date Last Updated:	
Last Updated by:	
Actor:	Sensor
Description:	When observations for a sensor are made, the sensor will begin to register a new value in it's datastream. This change will need to be acknowlged by the microcontroller and read into the program used to format and send observations to the database.
Preconditions:	1. Sensor is connected to micro-controller
	2. Sensor is uniquely identified, and datastream has been created
	3. Observations are being collected
Postconditions:	1. Program running on micro-controller receives new observation
Priority:	High
Frequency of Use:	High
Normal Course of Events:	1. Observation is collected by sensor
	2. Micro-controller reads new observation
Alternative Courses:	
Exceptions:	2.ex Micro-controller does not read new observation
	2.ex.1 Micro-controller reads next observation, ignoring the lost observation
Includes:	1.1 Uniquely identify sensor
Special Requirements:	
Assumptions:	
Notes and Issues:	Much of this may change as the group continues working with sensors.

Appendix B: Website Use Cases

Use Case ID:	2.1
Use Case Name:	Assign a location to a microcontroller
Created by:	BT
Date Created:	24-Nov-13
Date Last Updated:	30-Nov-13
Last Updated by:	BT
Actor:	Person setting up or changing sensor configuration
Description:	The microcontroller will have no way of knowing it's geographic location, so this should be configurable by a person. Once connecting a microcontroller, the user should be able to see that a new 'thing' has been added to the sensing network (Do they register themselves or do they need to be registered?). This will have a unique ID and can be assigned a location based on a map of the store.
Preconditions:	1. User is logged in/authenticated
	2. System describing spatial location of shelves exists (may include map)
	3. Microcontroller has registered as a 'thing' in the database
Postconditions:	1. Entity type: Location is set
	2. New/no-location 'thing' flag removed
Priority:	Low
Frequency of Use:	High during initial set-up, low afterwards
Normal Course of Events:	1. User is made aware of 'thing' that has no location
	2. User enters a mode where the location of the 'thing' can be added (either via map or manual entry)
	3. User is shown location on map to verify that location is correct
	4. Location is saved to database
	5. Sensor's location can be accessed/displayed
Alternative Courses:	1.a 'Thing' has a location, but is being moved to a new location
	1.a.1 Remove Location attribute of associated sensors
	1.a.2 Remove location of microcontroller
Exceptions:	2.ex User enters a location that is not valid
	2.ex.1 User is notified of the error and asked for a new location
	4.ex Database is not available
	4.ex.1 User is notified of the error
Includes:	2.8 Log in/authenticate
	2.2 Configure a store layout
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	2.2
Use Case Name:	Configure a store layout
Created by:	BT
Date Created:	24-Nov-13
Date Last Updated:	30-Nov-13
Last Updated by:	BT
Actor:	High during initial set-up, medium during operation
Description:	There needs to be a way for a user to construct a store layout system in order to locate sensors. This will be based on a typical grocery store shelf. Attributes can be added to increase the specificity of the model, but a bare-bones requirement will be enforced to ensure data can be mapped to the right area. This system will work with a visual representation (map) to provide context and help users.
Preconditions:	1. User is logged in/authenticated
Postconditions:	1. Store layout is saved
	2. Spatial reference system can be used for other use-cases
Priority:	Medium
Frequency of Use:	Medium
Normal Course of Events:	1. User enters store layout creator
	2. User changes aisle configuration
	3. User changes facing configuration
	4. Spatial reference system is updated
Alternative Courses:	
Exceptions:	
Includes:	2.8 Log in/authenticate
Special Requirements:	
Assumptions:	Some sort of quad-tree data structure can accommodate the addition/removal/editing of shelve layout
Notes and Issues:	Functionality of this feature may need to be limited due to complexity, experience and time-restraints. In the worst case scenario, the website only features one spatial reference system that cannot be changed.

Use Case ID:	2.3
Use Case Name:	Get aisle traffic observations
Created by:	BT
Date Created:	24-Nov-13
Date Last Updated:	Novemeber 30,2013
Last Updated by:	BT
Actor:	Internal, part of 2.5 See aisle traffic
Description:	Observation data from the motion sensors needs to be accessible so that it can be displayed on the map. This data is located in the database, where each sensor will have a unique ID, associations with other objects, and other attributes. Using these associations to filter, sensor data needs to be requested with jquery.
Preconditions:	1. User is logged in/authenticated.
	2. A 'thing' with motion sensors exists in the database
	3. A datastream belongs to said 'thing', made from observations belonging to said sensor
Postconditions:	1. Data from motion sensor is retrieved from the database
Priority:	High
Frequency of Use:	High
Normal Course of Events:	1. Webpage gets new data from database using jquery
	2. Data is assigned to variable which can be used to display data
Alternative Courses:	1.a No new data exists
	1.a.1 No new data is assigned to variable
Exceptions:	
Includes:	
Special Requirements:	Each motion sensor observation is associated with a micro-controller from which it originated. This will be used to determine the location to show the data on the map.
Assumptions:	
Notes and Issues:	

Use Case ID:	2.4
Use Case Name:	Get facing observations
Created by:	BT
Date Created:	30-Nov-13
Date Last Updated:	
Last Updated by:	
Actor:	Internal, part of 2.6 See if products are faced
Description:	Observation data from the magnetometer/(other sensor?) needs to be accessible so that it can be displayed on the map. This data is located in the database, where each sensor will have a unique ID, associations with other objects, and other attributes. Using these associations to filter, sensor data needs to be requested with jquery.
Preconditions:	1. User is logged in/authenticated.
	2. A 'thing' with a sensor used for facing detection exists in the database
	3. A datastream belongs to said 'thing', made from observations belonging to said sensor
Postconditions:	1. Data from sensor used for facing detection is retrieved from the database
Priority:	High
Frequency of Use:	High
Normal Course of Events:	1. Webpage gets new data from database using jquery
	2. Data is assigned to variable which can be used to display data
Alternative Courses:	1.a No new data exists
	1.a.1 No new data is assigned to variable
Exceptions:	
Includes:	
Special Requirements:	Each facing detection sensor observation is associated with a micro-controller from which it originated. This will be used to determine the location to show the data on the map.
Assumptions:	
Notes and Issues:	

Use Case ID:	2.5
Use Case Name:	See aisle traffic
Created by:	BT
Date Created:	30-Nov-12
Date Last Updated:	
Last Updated by:	
Actor:	Manager, Store Clerk
Description:	A user (Manager or store clerk) should be able to see if the aisle traffic on a map of the store. This should update automatically without input from the user, so they can have an idle screen showing the data as it become available. If they wish, this data can be filtered in some useful ways. It will have both a graphical and numerical version of the data, displayed in an attractive and easy to understand format.
Preconditions:	1. User is logged in/authenticated.
	2. A 'thing' with motion sensors exists in the database
	3. A datastream belongs to said 'thing', made from observations belonging to said sensor
	4. Aisle traffic observations are being retrieved by the database and assigned to a variable
Postconditions:	1. Traffic information is shown on the map in a graphical form
	2. Traffic information is shown on the map in numerical form
Priority:	Medium
Frequency of Use:	High
Normal Course of Events:	1. User navigates to map view
	2. Observations are shown without input from user
Alternative Courses:	2.a User adds filter input
	2.a.1 Data displayed based on the filters that the user has configured
	2.b There are no observations
	2.b.1 There is an indication that the sensor is working, but that it is in an idle state
Exceptions:	2.ex Observations are not being shown on map
	2.ex.1 User is shown an error message of some kind, most likely propagated from use case 2.4 get aisle traffic observations
Includes:	2.8 Log in/authenticate
	2.1 Assign a location to a micro-controller
	2.2 Configure a store layout
	2.3 Get aisle traffic observations
	2.7 Show map UI
Special Requirements:	
Assumptions:	

Use Case ID:	2.6
Use Case Name:	See if products are faced
Created by:	BT
Date Created:	30-Nov-12
Date Last Updated:	
Last Updated by:	
Actor:	Manager, Store Clerk
Description:	A user (Manager or store clerk) should be able to see if the products are faced in an area of the store via the map UI. This should update automatically without input from the user, so they can have an idle screen showing the data as it become available. If they wish, this data can be filtered in some useful ways. It will have both a graphical representation of the data, displayed in an attractive and easy to understand format.
Preconditions:	1. User is logged in/authenticated.
	2. A 'thing' with a sensor used for facing detection exists in the database
	3. A datastream belongs to said 'thing', made from observations belonging to said sensor
	4. Facing information observations are being retrieved by the database and assigned to a variable
	5. Store layout has been configured
Postconditions:	1. Facing information is shown on the map in a graphical form
Priority:	Medium
Frequency of Use:	High
Normal Course of Events:	1. User navigates to map view
	2. Observations are shown without input from user
Alternative Courses:	2.a User adds filter input
	2.a.1 Data displayed based on the filters that the user has configured
Exceptions:	2.ex Observations are not being shown on map
	2.ex.1 User is shown an error message of some kind, most likely propagated from use case 2.5 Get facing observations
Includes:	2.8 Log in/authenticate
	2.1 Assign a location to a micro-controller
	2.2 Configure a store layout
	2.4 Get facing observations
	2.7 Show map UI
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	2.7
Use Case Name:	Show Map UI
Created by:	BT
Date Created:	30-Nov-13
Date Last Updated:	
Last Updated by:	
Actor:	Internal / User (manager, store clerk)
Description:	Once a store layout is configured, there should be a map UI where the user can see observation data in it's correct location. This map will include controls to filter observations in certain ways. The map should update as new observations become available.
Preconditions:	1. User is logged in/authenticated
	2. Store layout has been configured
Postconditions:	1. Map is shown
Priority:	Medium
Frequency of Use:	High
Normal Course of Events:	1. User navigates to map view
	2. Map is shown
Alternative Courses:	
Exceptions:	2.ex Map cannot be rendered from store layout
	2.1.ex Error message is shown
Includes:	2.8 Log in/authenticate
	2.2 Configure a store layout
Special Requirements:	Map enables 2.5 and 2.6 to be displayed. The map will need to update automatically
Assumptions:	
Notes and Issues:	

Use Case ID:	2.8
Use Case Name:	Log In/authenticate
Created by:	BT
Date Created:	30-Nov-13
Date Last Updated:	
Last Updated by:	
Actor:	User (Store clerk, Manager
Description:	In order to protect a group's data, authentication will be required to access an account. Depending on if the user is a store clerk or Manager, they can have different permissions.
Preconditions:	1. Account exists with username and password
Postconditions:	1. User is logged in and presented with their group's data
Priority:	Low
Frequency of Use:	High
Normal Course of Events:	1. User enters username and password
	2. Username password combo is verified and user is logged in
Alternative Courses:	
Exceptions:	2.ex Username password combo is incorrect
	2.ex.1 User is not logged in, and informed that they entered the wrong User/pass
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	There will be a different level of permission for a manager and store clerk

Use Case ID:	2.9
Use Case Name:	Become System configurator
Created by:	BT
Date Created:	30 November, 2013
Date Last Updated:	
Last Updated by:	
Actor:	Manager
Description:	If a manager logs in, they should have the powers of a system configurator. This will allow them to change store/micro-controller layout properties.
Preconditions:	1. Manager is logged in
	2. Permissions are set
Postconditions:	1. Manager has access to store layout
	2. Manager can assign a location to a micro-controller
Priority:	Medium
Frequency of Use:	Low
Normal Course of Events:	1. Manager logs in
	2. Manager navigates to a configuration options section/page
	3. Store layout and micro-controller location are edit-able
Alternative Courses:	
Exceptions:	
Includes:	2.8 Log in/autenticate
	2.2 Configure a store layout
	2.1 Assign a location to a microcontroller
Special Requirements:	
Assumptions:	
Notes and Issues:	

References

[1] Margaret Rouse, “What is the Internet of Things (IoT)?,” *whatis.com*, Jul-2013. [Online]. Available: http://whatis.techtarget.com/definition/Internet-of-Things. [Accessed: 05-Dec-2013].

[2] “Here’s Why ‘The Internet of Things’ Will Be Huge, And Drive Tremendous Value for People And Businesses,” *Growth In The Internet of Things - Business Insider*, 22-Nov-2013. [Online]. Available: http://www.businessinsider.com/growth-in-the-internet-of-things-2013-10. [Accessed: 05-Dec-2013].

[3] “OGC SensorThings API,” *OGC SensorThings API*, 2013. [Online]. Available: http://ogc-iot.github.io/ogc-iot-api/. [Accessed: 05-Dec-2013].

[4] Mirela Popa, Leon Rothkrantz, Zhenke Yang, Pascal Wiggers, Ralph Braspenning, and Caifeng Shan, “Analysis of Shopping Behavior based on Surveillance System,” in *Analysis of Shopping Behavior based on Surveillance System*, 2010, pp. 2512–2519.

[5] Pierre Chandon, J. Wesley Hutchinson, Eric T. Bradlow, and Scott H. Young, “Does In-Store Marketing Work? Effects of the Number and Position of Shelf Facings on Brand Attention and Evaluation at the Point of Purchase,” *J. Mark.*, vol. 73, pp. 1–17, 2009.

[6] Object Modeling Group, “OMG Unified Modeling Language InfraStructure,” *UML 2.2*, Feb-2009. [Online]. Available: http://www.omg.org/spec/UML/2.2/. [Accessed: 05-Dec-2013].

[7] Object Modeling Group, “OMG Unified Modeling Language SuperStructure,” *UML 2.2*, Feb-2009. [Online]. Available: http://www.omg.org/spec/UML/2.2/. [Accessed: 05-Dec-2013].

[8] “PIR Motion Sensor,” *SK Pang Electronics, Arduino, Sparkfun, GPS, GSM*. [Online]. Available: www.skpang.co.uk/catalog/index.php. [Accessed: 05-Dec-2013].

[9] Alex Louden, “Mechatronics Project - Sensors,” *Alex Louden*, 09-Aug-2010. [Online]. Available: http://alexlouden.com/2010/mechatronics-project-sensors/. [Accessed: 05-Dec-2013].

[10] Kathleen Ang, Ben Trodd, Jeremy Steward, Alexandra Cummins, and Harshini Nanduri, “ENGO 500: GIS & Land Tenure #2 - Literature Review: Location Aware Smart Shelves,” University of Calgary, Literature Review 2, Nov. 2013.

[11] “Bootstrap,” *GetBootstrap*. [Online]. Available: http://getbootstrap.com/2.3.2/. [Accessed: 05-Dec-2013].

[12] “jQuery - Write less, do more.,” *jQuery*. [Online]. Available: jquery.com. [Accessed: 05-Dec-2013].

[13] “D3: Data-Driven Documents,” *D3.js - Data-Driven Documents*. [Online]. Available: http://d3js.org/. [Accessed: 05-Dec-2013].

[14] Jeremy Steward, “ThatGeoGuy/ENGO500,” *ThatGeoGuy/ENGO500*. [Online]. Available: https://github.com/ThatGeoGuy/ENGO500. [Accessed: 05-Dec-2013].

[15] Kathleen Ang, Ben Trodd, Jeremy Steward, Alexandra Cummins, and Harshini Nanduri, “ENGO 500: GIS & Land Tenure #2 - Project Proposal,” University of Calgary, Proposal 1, Sep. 2013.