SENSOR DATA PLATFORM WEB APPLICATION

Technical Report

Project 3

Fall, 2013

Arie Radilla Laureano

Isil Demir

Lyman Cao

Yazhisai Gowthaman

Carnegie Mellon University

SILICON VALLEY

Table of Contents

1 Executive Summary

We have attempted to rewrite the current Ruby on Rails web app on Heroku (http://cmu-sds.herokuapp.com/) into Java Play framework, aligning with the current APIs, to handle metadata view/creation/deletion of our Sensor Data Service Platform. Also, this platform will allow users to report bugs and have a detailed view of active vs. non-active sensors. This platform makes use of the Sensor Service Platform API, a REST API for interacting with the platform.

2 Project Background

Sensors have become a predominant part of digital interaction with the physical world. Managing and viewing these sensors is an integral part of accessing and using their data. In an attempt to simplify accessing this sensor data, we have come up with the Sensor Data Platform Web Application.

The application aims to provide a platform for the general public to access the sensor data. By means of this application, user can view sensors, devices they are attached to, their locations and the sensor data. In addition, users can also see a snapshot of all available sensors with accurate information of active and inactive sensors at any given time. Apart from providing a platform to view sensor related information, the application also acts as a channel for users to provide feedback to CMU.

Our vision is to develop this web application with special focus to enable to it to be independent and extensible to handle higher loads (in terms of access) and open to a larger public as a one-stop point for relevant sensor data.

3 Architectural Design

The architecture design of this project relies on the metadata to demonstrate the important components.

3.1 Metadata

The metadata is the core of the application. All relevant data about the Sensors and the sensor data are represented in the database as the metadata. Below is a schema of the metadata for the application.

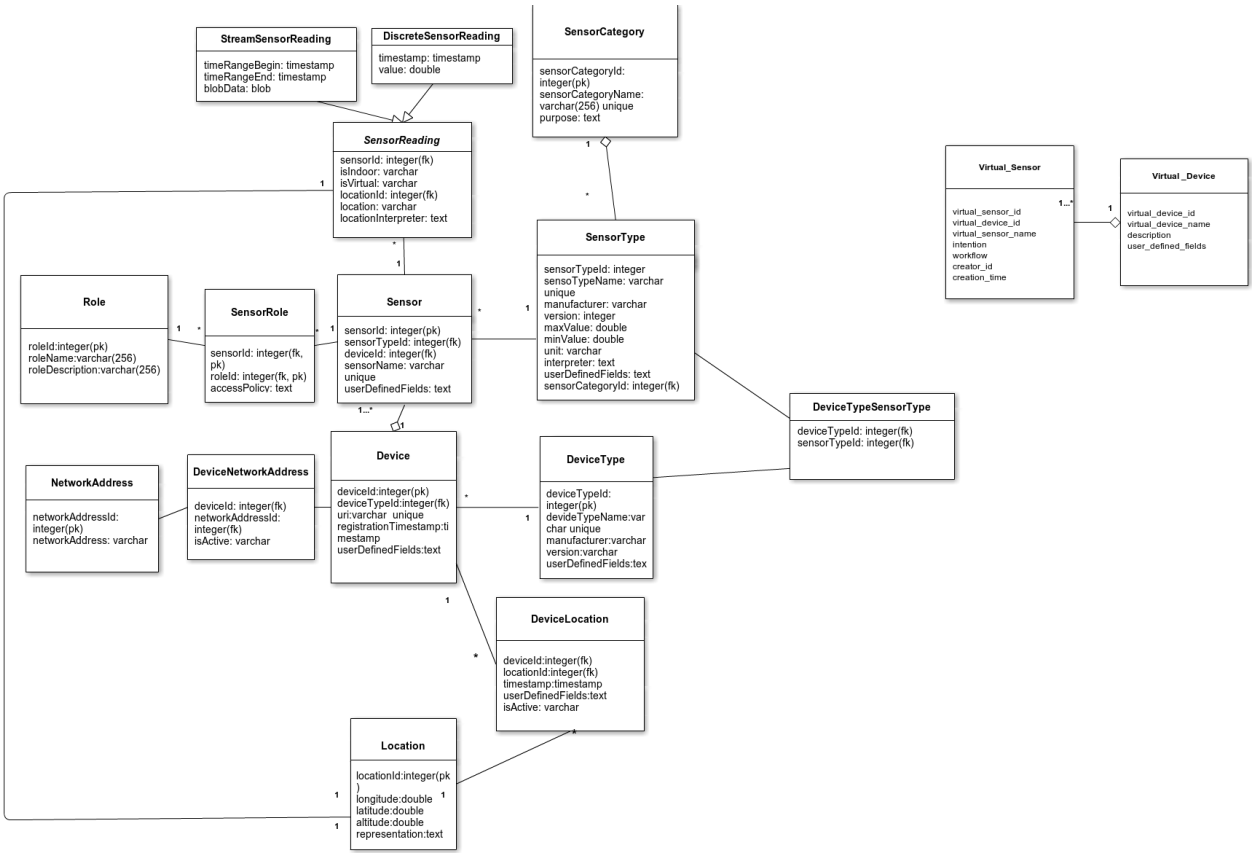


Figure 1 - Metadata

4 ATAM Architecture Analysis

Below is the ATAM analysis for our application

4.1 Architectural Approaches

Listed in this section are the architectural approaches utilized in the Sensor Data Platform web site. They were picked and implemented for the primary reason of focusing on quality attributes such as modifiability, security, performance, extensibility and reliability.

4.1.1 MVC

The Model-View-Controller paradigm allows the web application to be modular, and developed rapidly. The separation of the view and the business logic allows multiple view implementations for a single model, as well as offering a flexible development environment where frontend and backend concerns are separated. This separation also ensures that the application is more secure and extensible making the application open to future enhancements.

4.1.2 RESTful

The web framework on which the website is built is entirely RESTful, which makes it very easy to scale the web application, as well as introducing the concept of caching client requests and greatly improves performance. REST's separation of client and server is especially important in increasing effectiveness of performance tuning and scalability of server components.

It is also widely practiced in web development, thus bringing the web application up to par with industry standards.

4.1.3 In-memory database

The database engine the web application uses for local data persistence can be configured as an in-memory database, which allows queries to run much faster, compared to tables stored on a disk.

4.1.4 Stateless

The application also follows a stateless implementation, meaning that the client states are never stored in the server. Having each application instances to be completely stateless means that client requests can be load balanced across all the available virtual or physical machines. This allows a perfect venue for scalability and portability.

4.1.5 Non-blocking I/O

The application's web framework allows the server to handle client requests asynchronously. That is to say, a client request can be processed while other processes continue to run in parallel, making the web application highly efficient in handling concurrent client requests.

Considering that the application is expected to expand and be used by a large number of users, this improved the overall efficiency.

4.1.6 Layered

The web application uses a completely separate set of API calls for its actual set of functionality. This allows the developers to easily differentiate between the web application and the API layer, in case of failures.

4.2 Utility Tree

* (H, L) means HIGH importance to system success, LOW difficulty in achieving

Quality Attribute	**Availability**
Attribute Concerns	Does not crash
Scenario(s)	(H, L) The application should not crash when the user inputs some

	wrong format data
Attribute Concerns	Graceful degradation in the presence of failures
Scenario(s)	(H, M) If the function of viewing dashboard fails, the user can still view and operate the metadata
Quality Attribute	**Privacy**
Attribute Concerns	Authentication before data writing
Scenario(s)	(H, L) The user needs be to authenticated before he/she writes on the data
Attribute Concerns	Certain sensitive data must be manipulated to protect source of information
Scenario(s)	(H, L) The database should store users' password encrypted
Quality Attribute	**Usability**
Attribute Concerns	Learnability
Scenario(s)	(M, L) The user can learn how to use this web application in a short period
Attribute Concerns	Memorability
Scenario(s)	(M, L) The user can memorize how to use this web application
Quality Attribute	**Performance**
Attribute Concerns	Latency
Scenario(s)	(M, M) The user should not wait longer than three seconds to load the page
Attribute Concerns	Transaction throughput

Scenario(s)	(L, L) The transaction throughput should at least support higher than 1 transaction / sec
Quality Attribute	**Extensibility**
Attribute Concerns	The system supports extension and takes into consideration future growth
Scenario(s)	(M, M) The system should handle addition of new features such as different methods of querying data
Quality Attribute	**Scalability**
Attribute Concerns	The system has the ability to accommodate growth
Scenario(s)	(M, M) The system should be able to increases in volumes either by provided resources or providing ease in adding new resources

4.3 Risks and Sensitivity Points

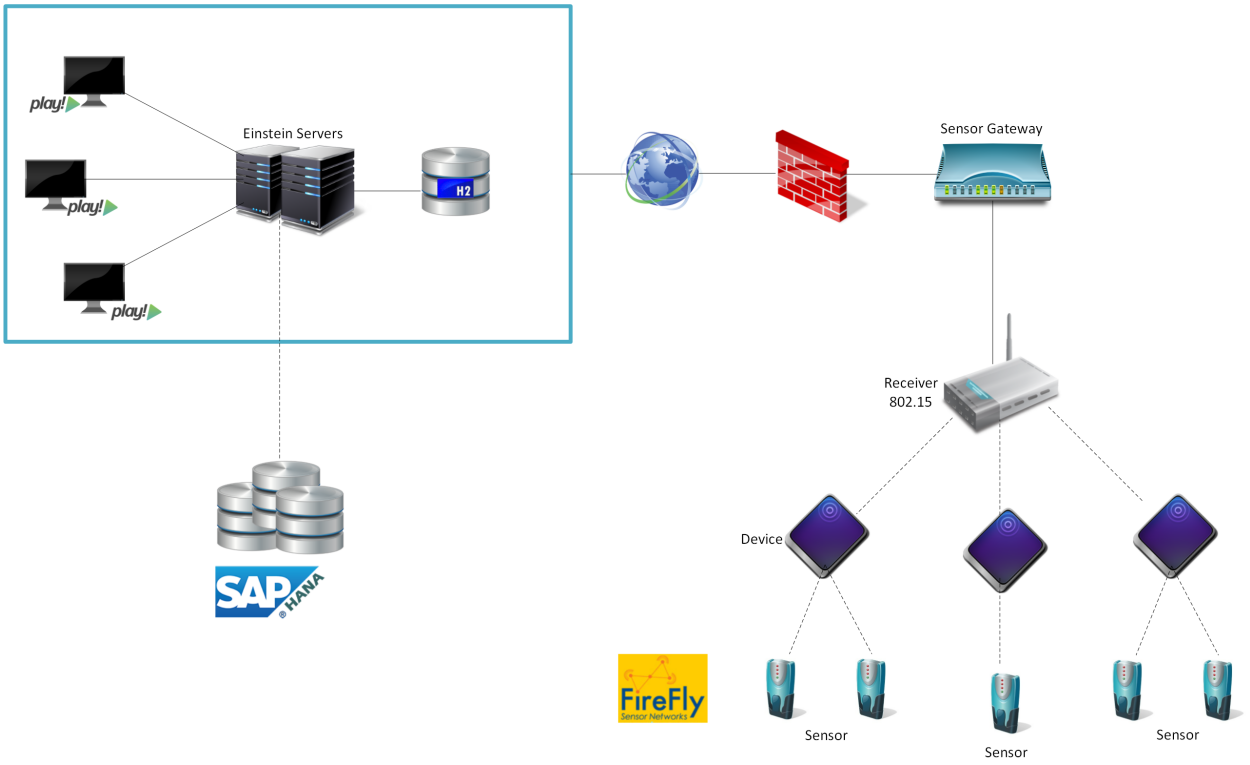
- Ensure requests performed by web application don't have a big impact on the federated sensors platform.
- Operations performed on the web application should not affect or modify the data generated by the platform.
- Architecture should allow for future growth, i.e. the accommodation of user-generated data, as well as current sensor readings.
- The most important component of this application is the Sensor Model, special care should be put to make this representation **as flexible and extensible as possible.**

5 Sensor Data Platform Web Application

The implemented prototype demonstrated the feasibility of the proposed architecture data model. Additionally it helped in understanding other complexities that may arise in implementation. Tradeoffs between factors such as computation and storage could also be better evaluated with a working model, which can then be incorporated into the design.

5.1 Design

The following architecture diagram demonstrates the components and communication of the Sensor Data Platform Web Application.



5.2 Additional Database Information

In addition to the HANA database where the primary information about sensors and relevant sensor is stored, the web application also uses an additional database for local storage of application specific data. This data includes user information and bug reports.

The application provides "Sign In" functionality to enable authorized users to perform additional operations such as deleting and creating sensor components and viewing bug reports. Such user information is stored in a local database – the H2 database engine.

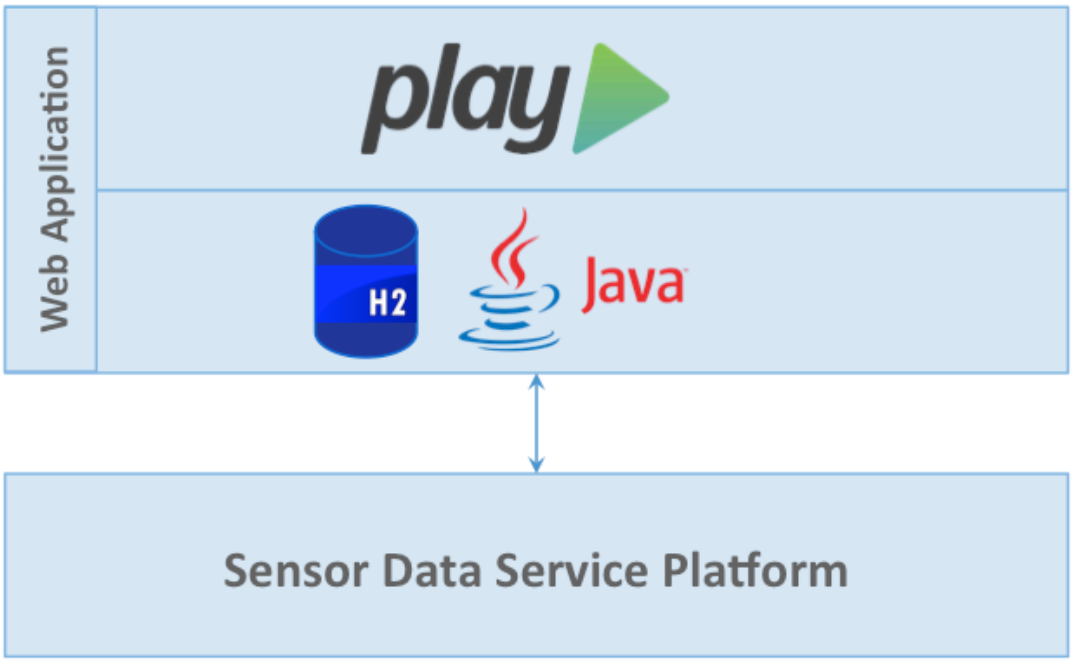
"Bug Reports" are also local to the application and is not a part of the Sensor Data Platform. This is also stored in the H2 database.

More information of H2 can be found at the below location.

http://www.h2database.com/html/main.html

5.3 Implementation

The web application is implemented to communicate with the Sensor Data Platform as represented in the diagram below.



5.4 Installation

- Install Play Framework. Instruction can be found at http://www.playframework.com/documentation/2.2.x/Installing
- Download the source code from https://github.com/cmusv-sc/ArchF2013-Project3-FT
- From the parent folder, run the command: 'play run'
- Local App will available at localhost:9000

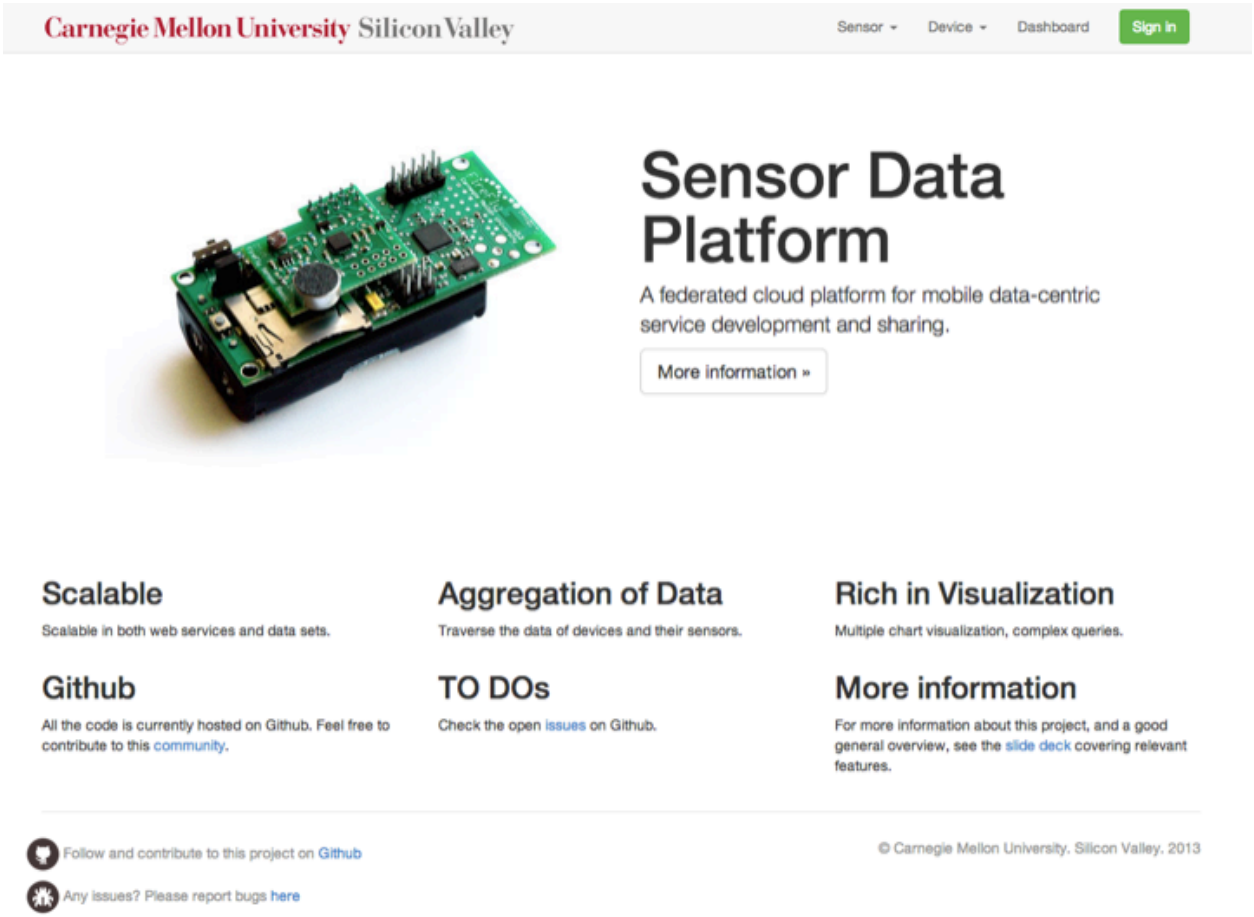
5.5 Tutorial

The Sensor Data Platform Web Application is hosted on Heroku and can be accessed through the below URL.

URL: http://fall-2013-cmusds.herokuapp.com/

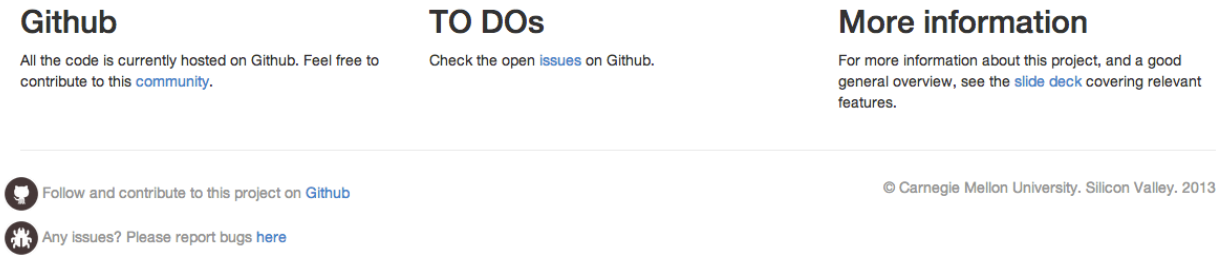
5.5.1 Home Page

The homepage of the application is very informative and self descriptive of the features and contents of the website



5.5.2 Useful links

The home page gives access to some of the useful links that provides various resources which assists in using the application.



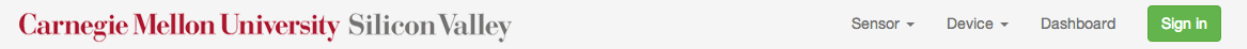
5.5.3 Menu

The menu at the top of the application provides links to all the key functionalities of the application

5.5.4 Sensor

Clicking on the “Sensor” menu and navigating to the “Sensors” sub menu takes the user to the page where all the sensors are listed.

Sensor >> Sensors



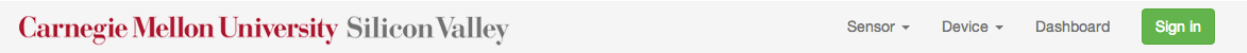
523 Sensor (s)

Sensor Name	Sensor Type	Device URI
acc_z	Accelerometer z	48
acc_z	Accelerometer z	53
acc_z	Accelerometer z	59
acc_z	Accelerometer z	65
temperature	Temperature_NASA	70
temperature	Temperature_NASA	71
acc_z	Accelerometer z	42
temp	Temperature_Jeenet	39
light	Light_Jeenet	39
humidity	14	39
sound_level	Sound_level_Jeenet	39
temp	Temperature_Jeenet	40
light	Light_Jeenet	40

5.5.5 Sensor Types

Clicking on the “Sensor” menu and navigating to the “Sensor Types” sub menu takes the user to the page where all the sensor types are listed.

Sensor >> Sensor Types



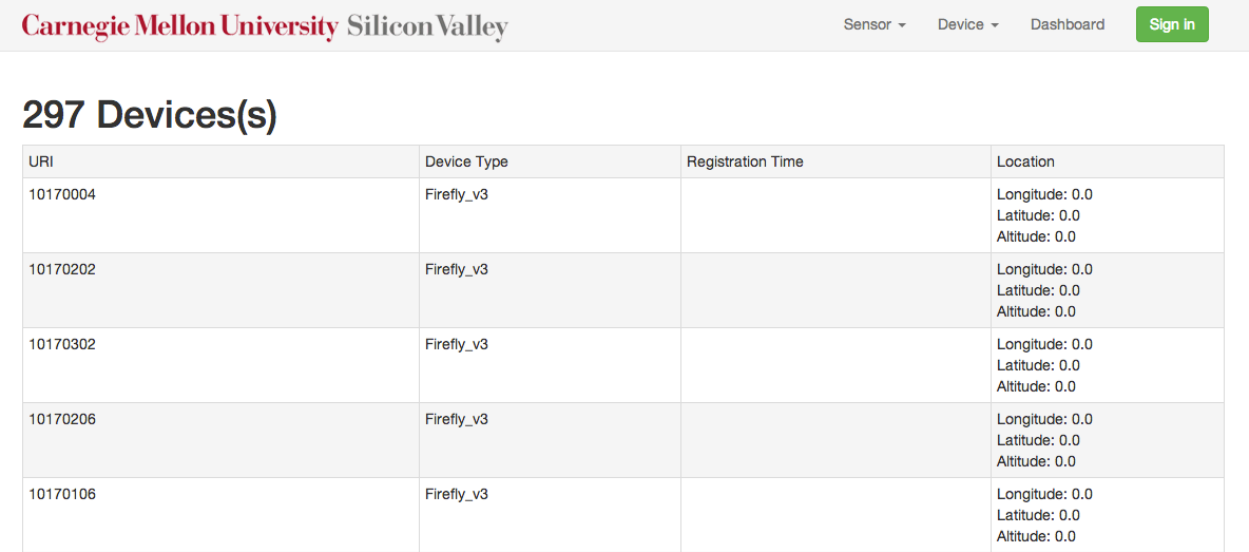
191 Sensor Type(s)

Sensor Type Name	Manufacturer	Version	Max Value	Min Value	Unit	Interpreter
Digital Temperature		0.0	0.0	0.0		
Light		0.0	0.0	0.0		
Humidity		0.0	0.0	0.0		
Motion		0.0	0.0	0.0		
Accelerometer x		0.0	0.0	0.0		
Accelerometer y		0.0	0.0	0.0		
Accelerometer z		0.0	0.0	0.0		
Audio P2P		0.0	0.0	0.0		
person_counter		0.0	0.0	0.0		
Light_Jeenet		0.0	0.0	0.0		
Temperature_Jeenet		0.0	0.0	0.0		
Sound_level_Jeenet		0.0	0.0	0.0		
Light_sweetfeedback		0.0	0.0	0.0		

5.5.6 Devices

Clicking on the “Device” menu and navigating to the “Devices” sub menu takes the user to the page where all the devices (to which the sensors are mapped) are listed.

Device >> Devices



5.5.7 Device Types

Clicking on the “Device” menu and navigating to the “Device Types” sub menu takes the user to the page where all the sensor types are listed.

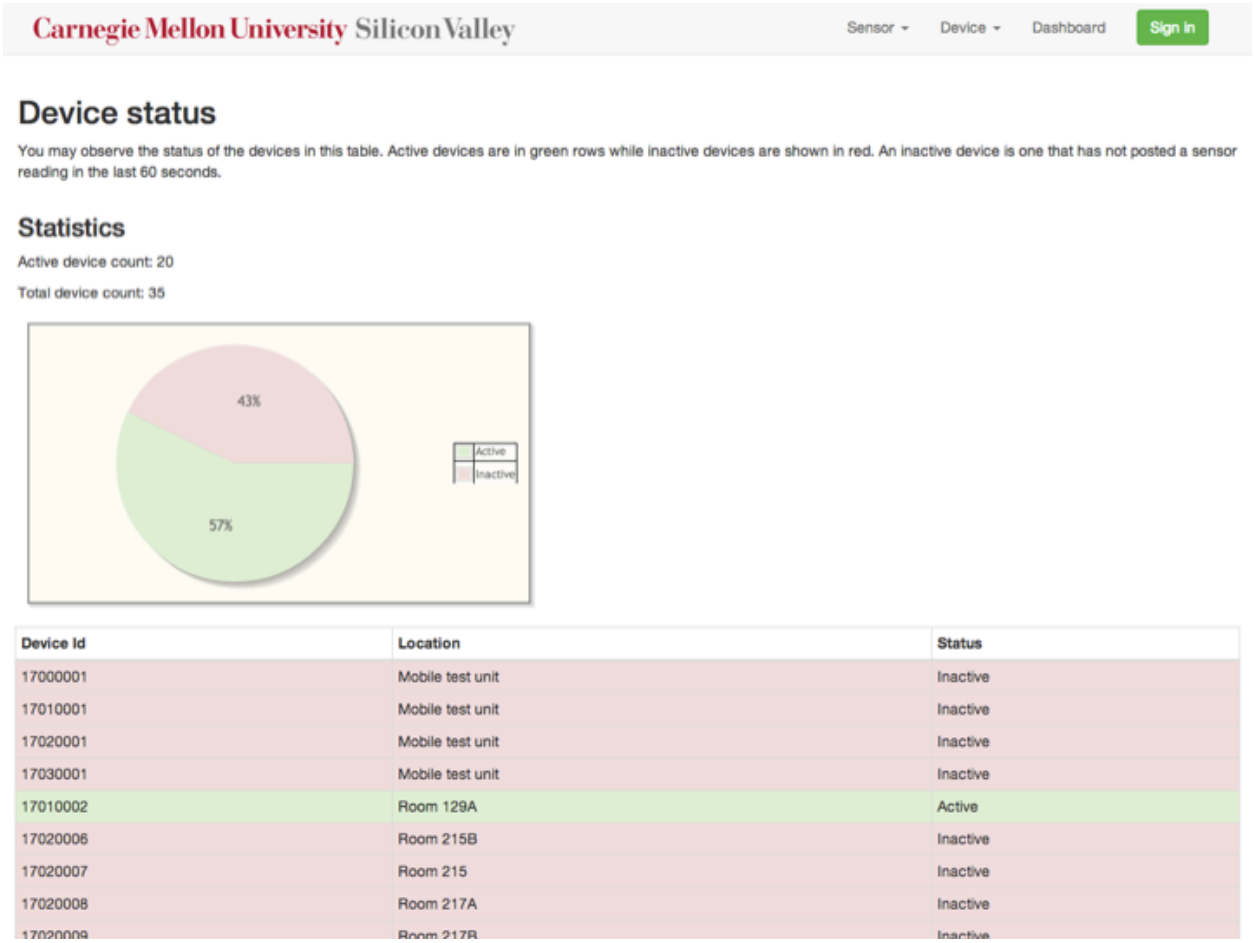
Device >> Device Types

142 Devices Type(s)

Device Type Name	Manufacturer	Version
ElectricImp	Spark Fun	1.0
Firefly_v2	CMU	1.0
Jeenet_node	Art	2.1
Sweetfeedback	Ted	1.0
Firefly_v3	CMU	1.0
NASA		-1.0
test_device_3	MG	1.0
test_device_3	MG	1.0
test_device_3	MG	1.1

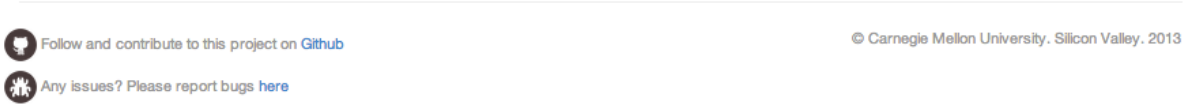
5.5.8 Dashboard

The Dashboard acts as a one-stop page to view all the active and inactive sensors and get a high level picture of the sensor activity.



5.5.9 Bug Report

Our application also provides an option for users to report bugs with the Bug Report feature, which can be found at the footer of all pages



Clicking on this link, takes the user to the Bug Report page as shown below.

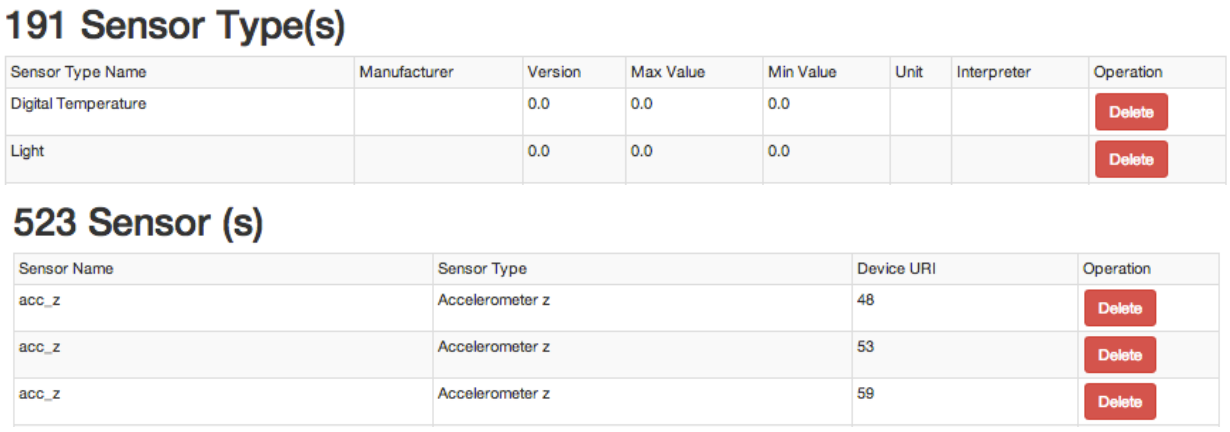
5.5.10 Sign In

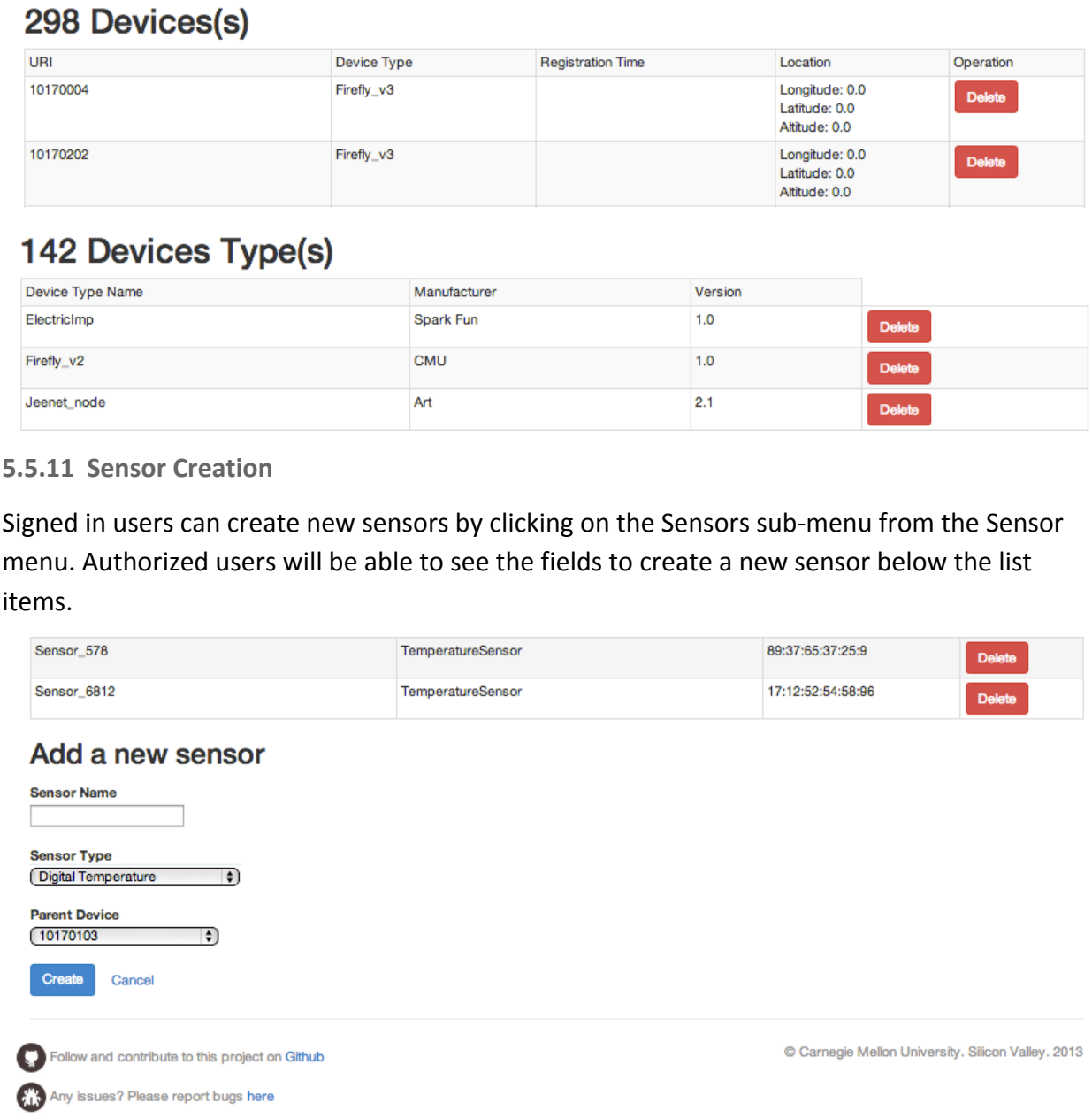
Apart from the above features, our application also provides authorized users with addition access as below.

Clicking on the sign-up link brings up the sign up page.

Signing in provides the additional features:

Option to Delete



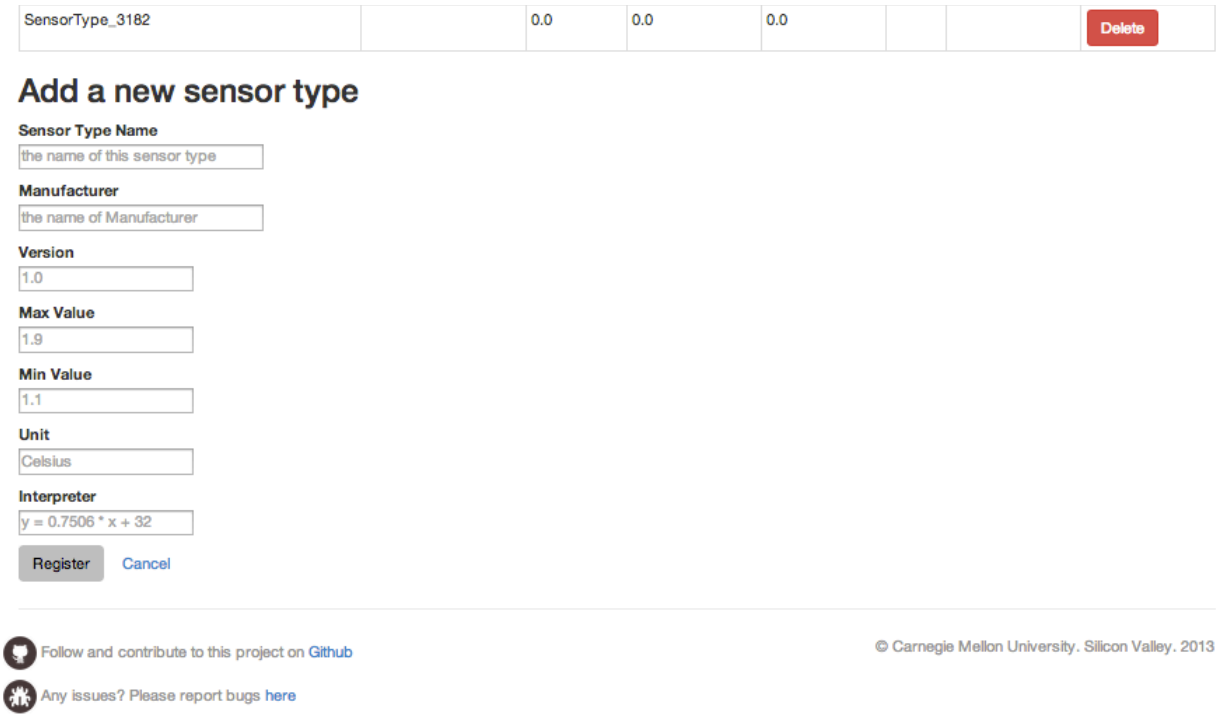


5.5.11 Sensor Creation

Signed in users can create new sensors by clicking on the Sensors sub-menu from the Sensor menu. Authorized users will be able to see the fields to create a new sensor below the list items.

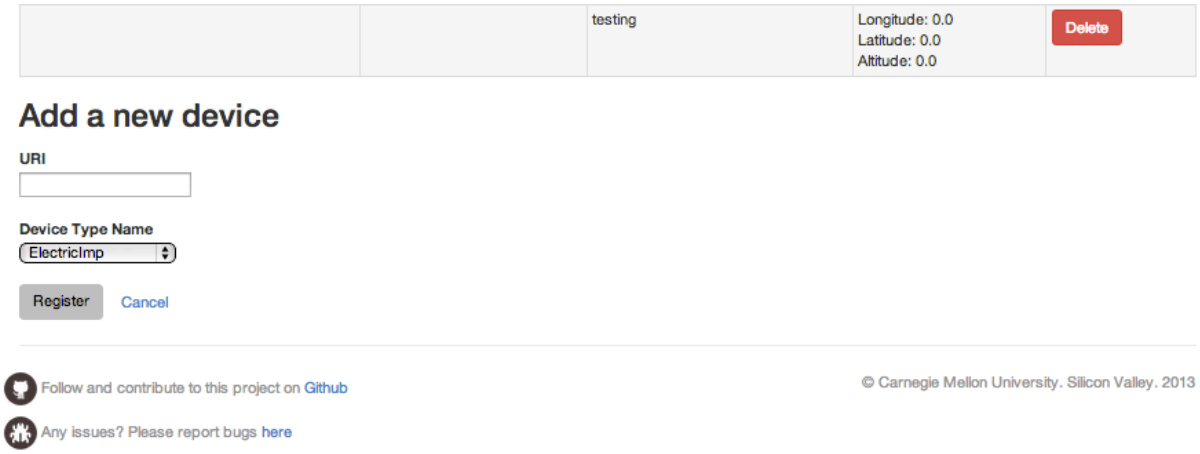
5.5.12 Adding a new Sensor Type

Authorized users can create a new Sensor Type by navigating to the Sensor Types page. They will be able to see the fields to create a new sensor type below the list items.



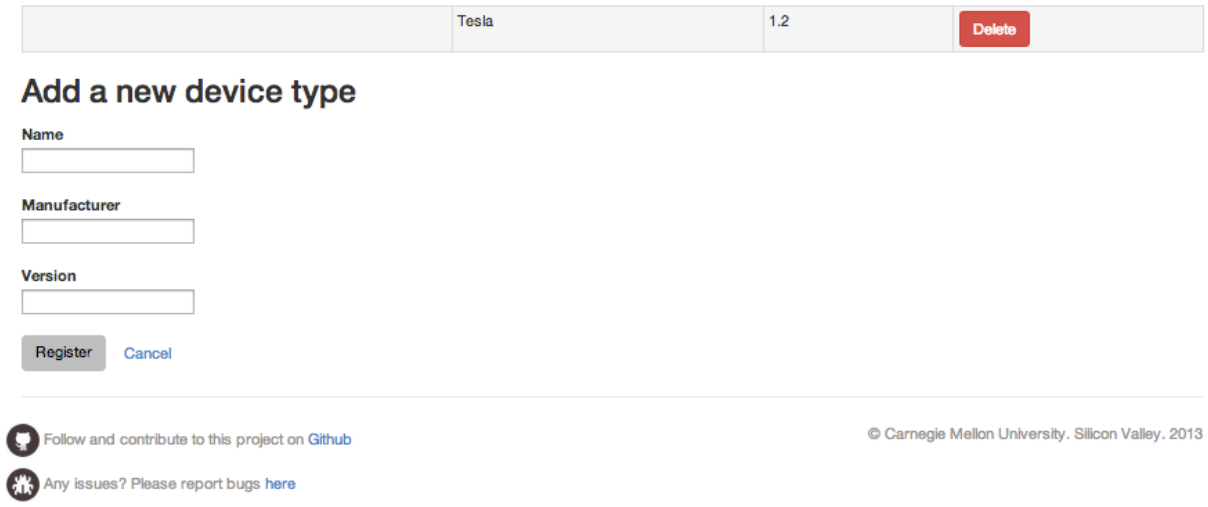
5.5.13 Create a new Device

Authorized users can create a new Device by navigating to the Devices page. They will be able to see the fields to create a new Device below the list items.



5.5.14 Create a new Device Type

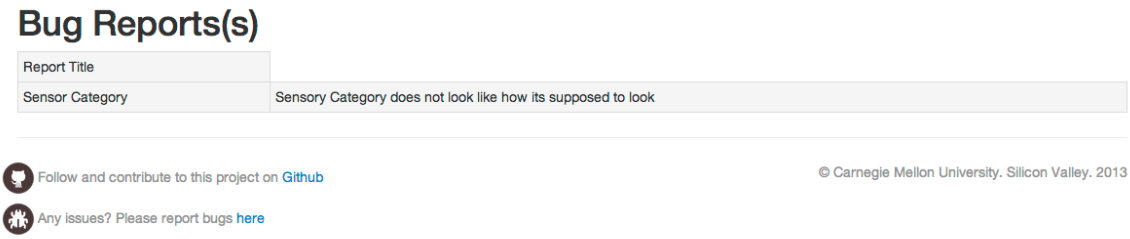
Authorized users can create a new Device Type by navigating to the Device Types page. They will be able to see the fields to create a new Device Type below the list items.



5.5.15 View Bug Reports

Authorized users can view logged bugs in addition to creating a new bug reports. Signed in users will be able to see an additional menu item under Bug Report to view the logged bugs.

Bug Report >> Reports



6 REST API

API calls for accessing the sensor data can be found at the below location:

https://github.com/cmusv-sc/ArchF2013-Project2-FT

Below is an example of the detailed usage of an API call to get the sensor readings of a sensor as mentioned in Sensor Service Platform APIs Version 1.3[1]

GET SENSOR READING FROM A SENSOR (SPECIFIED BY SENSOR NAME) AT A TIMESTAMP

[1] Detailed Usage of API called taken from https://github.com/cmusv-sc/ArchF2013-Project2-FT

- **Purpose**: Query sensor reading for a specific sensor at a specific time point.
- **Method**: GET
- **URL**: http://einstein.sv.cmu.edu:9000/getSensorReading/<"sensorName">/<"timestamp">/<"resultFormat">
- **Semantics**:
  - **sensorName**: Existing sensor name.
  - **timestamp**: Time of the readings to query.
  - **resultFormat**: Either JSON or CSV.
- **Sample Usages**:
  - **Sample csv request**: http://einstein.sv.cmu.edu:9000/getSensorReading/sensor1/1368568896000/csv
  - **Sample csv result**: (sensorName,timestamp,value) sensor1,1368568896000,518.0
  - **Sample json request**: http://einstein.sv.cmu.edu:9000/getSensorReading/sensor1/1368568896000/json
  - **Sample json result**:

    {"sensorName":"sensor1","isIndoor":true,"timeStamp":"Dec 31, 1969 7:36:40 PM","locationInterpreter":"jdkfdj","value":"50"}
  - **Result**: HTTP 200 if returned successfully, HTTP 404 if not found.

7 Source Code

The source code of this application is available on Github at the following location:

https://github.com/cmusv-sc/ArchF2013-Project3-FT

Default Login Credentials

Username: admin@admin.com

Password: adminadmin

8 Future Work

The team faced some challenges, which resulted in a few features to not be implemented. The challenges and features are listed below.

- Historic data cannot be deleted due to inconsistency in the new data model and the old data model
- Performance issues - when the retrieved data is huge, the application take a long time to load
- Missing APIs - Some of the APIs like Edit APIs and Sensor Category are currently unavailable due to which these functionalities have not been implemented

Future work includes implementation of these missing features.

9 Contact Information

Find below, the contact information of the team members involved in developing the application.

Lyman Cao
lyman.cao@sv.cmu.edu
650-933-7515

Isil Demir
isil.demir@sv.cmu.edu
650-996-1201

Arie Radilla Laureano
arie.radilla.laureano@sv.cmu.edu
323-804-4349

Yazhisai Gowthaman
yazhisai.gowthaman@sv.cmu.edu
408-386-4243