Integrating Various Building Metrics from the AMS Student Nest into a Presentable Dashboard

Anna Gudimova, Dan Lee, Euan Chow, Mike Griffin, Yuxin Xu

University of British Columbia

EECE 409/429/419/439/400/469

April 08, 2016

The University of British Columbia

Final Report Including:

Requirements Specification, Design, Verification & Validation, and Suggestions For Project Continuation Reports

Integrating Various Building Metrics from the AMS Student Nest into a Presentable Dashboard (Campus + Community Planning, AMS Sustainability)

Prepared By:

Anna Gudimova

Euan Chow

Meng-Yeng (Dan) Lee

Micheal Griffin

Yuxin (Eugene) Xu

UBC AMS Sustainability

April 8, 2016

The following is a list of reports that identify the requirements for the implementation, the design decisions satisfying the requirements, the verification and validation of the system and finally a few suggestions for project continuation by our client.

The University of British Columbia

Requirements Specifications V1.0

Integrating Various Building Metrics from the AMS Student Nest into a Presentable Dashboard (Campus + Community Planning, AMS Sustainability)

Prepared By:

Anna Gudimova

Euan Chow

Meng-Yeng (Dan) Lee

Micheal Griffin

Yuxin (Eugene) Xu

UBC AMS Sustainability

April 7, 2016

Executive Summary

With the recent construction of the University of British Columbia's (UBC) Alma Mater Society (AMS) Student Nest building, the AMS Sustainability office wants to integrate it into UBC's culture of environmental consciousness. Sensors to collect building data were installed throughout the Nest, an in-vessel composter and waste scale was installed, and databases were created to store and aggregate the information collected. However, because these databases are stored separately, comparing and visualizing this data using specific time intervals is needed so that the data can be more clearly analyzed and interpreted.

The goal of AMS Sustainability with regards to this project is to provide students with information about how much waste they produce in the Nest, and use that information to spread awareness and help cut down on garbage. Without going over-budget and protecting the security and integrity of the data collected, AMS Sustainability wants to design and create a new database on an active server that integrates the current databases into one location.

Table of Contents

Revision History

Author	Date	Reason For Changes	Version
Micheal Griffin	**10/30/2015**	Proposal for Database	**V1.0**
Anna Gudimova	**11/16/2015**	Change from Mongo DB to Microsoft SQL	**V1.1**
Anna Gudimova	**11/30/2015**	Changes to the frequency of data requests	**V1.2**
Micheal Griffin	**02/15/2016**	Removing the dashboard implementation as part of the requirements for this project	**V1.3**

1 Introduction

1.1 Product Overview

This document outlines the functional and nonfunctional requirements and constraints for the AMS Nest Dashboard, which includes the integration of the two main sources of data in the building onto one Canadian server; and the implementation of the dashboard using Visual Studio to display building metrics in the Nest. This is the first release (version 1.3) of the dashboard.

2 Specific Requirements

The following is a list of the requirements that need to be met in order to satisfy the clients needs, and allow for further refinements. Any constraints in relation to the requirements are included, and help to identify obstacles in completing the final deliverable. The list is used as a basis for our design decisions along with our testing and verification of the system.

2.1 External Interface Requirements

2.1.1 User Interfaces

The dashboard must have all waste and energy sensor data recorded and available for research purposes. The dashboard must display the data in a recognizable format so that users are capable of using the data for their own projects. The categories must be consistent with the themes currently used in the AMS Nest, and UBC.

The system must not display inaccurate or “zero” results when calculating the difference in consumption of energy, or production of waste. As an example, if the gas sensors produce negative values, we must not display this to the end-user. That is, someone interacting with the dashboard should only be given improvement statistics or retrogressions based on the sensor data from the Waste Scale, or the AMS Sensor Database.

2.1.2 Software Interfaces

The Canadian Cloud Server must have Python 2.7 (DO NOT upgrade Python) and pymssql library installed. All three nodes of the back end, the Cloud Server, the AMS Sensor Server, and the Waste Scale Computer must always be on and must have internet connection. If python script files need to be relocated, they must be moved along with the whole folder in which they are located. The batch file that runs the python script must not be removed from the Startup folder on either the Cloud Server or the Waste Scale computer.
The database on the Cloud Server must only store positive or zero resource consumption measurements.

2.1.3 Communication Protocols

The AMS Sensor Server and the Waste Scale computer must have port 1433 opened for TCP connection to the Canadian Cloud server.

2.2 Functional Product Features

The following list outlines the functional requirements for the Nest Dashboard. These requirements were created based on client needs, project scope, and the restrictions related to the implementation of the dashboard and the database.

2.2.1 Database Requirements

1. Integrate two of the three main building metrics in the Nest into one location: the building aggregate data, which includes sensors collecting information for greywater, electricity, water usage, natural gas, solar hot water and district energy; and the waste scale, tallying the total amount of solid waste, recycling, paper and organics exiting the Nest daily.
    a. The in-vessel composter - storing information about soil production from in-house compost - was temporarily removed from the requirements of this deliverable because the sensors were broken. The data will be integrated at a later time.
2. The database must be on an accessible server with minimal delay (unnoticeable to users).
3. Information stored on the database must date back one year before being transferred to a local computer in the Nest.
4. Integrated database must verify the information being mirrored by the AMS Sensor database, and verify the results.
5. The layout must be modifiable in the future.
    a. For example, if a new sensor measuring something like building occupancy is added to the Nest, it must be possible to create a new table in the database without having to rebuild it.
6. The database must be able to take multiple requests without failing, and ensure atomicity.
    a. An example is to say that if 100 data requests are being written to the integrated database, a method should ensure that either all of the data has accurately been sent, otherwise decline the request and confirm none of it has been received to ensure the database avoids missing information or duplicates.
7. The units for each sensor must be listed in each table, and be consistent.

2.2.2 Dashboard Requirements

1. The dashboard must address two main features:
    a. Display trends related to student behaviour and their impact on the Nest

i. For example, the amount of waste created from one week to another.

b. Display graphs and/or lists for research purposes.

i. Students must be able to capture the information for their own research.

2. The goal of the dashboard is to affect student behaviour. Research should back design decisions and inform the layout in order to encourage students to maintain sustainable patterns within the building.

2.3 System Attributes

2.3.1 Reliability

The database must track errors so that anyone interacting with the data can diagnose the problem. If the data is unavailable, the dashboard must be made aware of the problem, so that atomicity can be relayed throughout the system. For example, if the server is unable to reach the waste scale, and the database is unable to record information in the Waste tables, the dashboard should avoid displaying information that may be inaccurate. This ensures the data is reliable.

2.3.2 Availability

The cloud server must poll the AMS Sensor Database and the Waste Scale Database so that information can be added once it is available. If either system is unreachable, the data must be added once it is back online.

2.3.3 Security

Confidentiality and integrity of the database must be protected, i.e. potential intruders must not be able to access or modify the data being transferred.

2.3.4 Maintainability

The database must allow for modifications in the future such as additional sensors, unit measurement changes (g to Kg), and maintain separate tables for each data type.

2.3.5 Portability

The software must be capable of running on a windows environment. The user interface must use software capable of being deployed on the dashboard in the Nest, and avoid using processing power above 1GHz, or more than 1Gb memory (constraints related to the dashboard). The language must be familiar to someone at a 4th year undergraduate level or higher, and have sufficient comments to make improvements and modifications for further development.

2.3.6 Performance

The dashboard must run on two screens simultaneously to address the need for user feedback and research capabilities. It must maintain feedback without delays, and avoid freezing or crashing without data present.

The University of British Columbia

Design Report

Integrating Various Building Metrics from the AMS Student Nest into a Presentable Dashboard (Campus + Community Planning, AMS Sustainability)

Prepared By:

Anna Gudimova

Euan Chow

Meng-Yeng (Dan) Lee

Micheal Griffin

Yuxin (Eugene) Xu

UBC AMS Sustainability

April 8, 2016

Table of Contents

1 Introduction

1.1 Design Overview

We will begin with introducing the top level design of the system. Illustration 1 displays a High-Level diagram of the project, and the components we will be addressing.

The system has two main components: the back end and front end. The back end resides on a Canadian Cloud server and contains SQL Server Express 2014 databases and data processing components called scripts written in Python 2.7. The front end - the Dashboard, is a Display that provides a user-friendly interface for analyzing and displaying data stored on the back end.

1.1.1 Back End

Back end integrates AMS sensor data for water, natural gas, and electricity measurements as well as AMS Waste Scale data for the amount of paper, recyclables, organics, and mixed garbage into two distinct SQL databases stored on a Canadian cloud server. Sensor data for each sensor is pulled from the mirror backup of the AMS Nest sensor database, downscaled to one entry, per day, per sensor, and inserted into the Sensor database on the cloud server. AMS Nest waste data is retrieved from text files stored on a local AMS Nest computer and inserted into the Waste database on the cloud server.

1.1.2 Front End

On the front end, C# application queries databases stored on the server, performs light data processing (for example, addition and comparison operations) and displays the data on our web page. This screen for the dashboard is located on the bottom floor of the NEST next to the AMS Sustainability center.

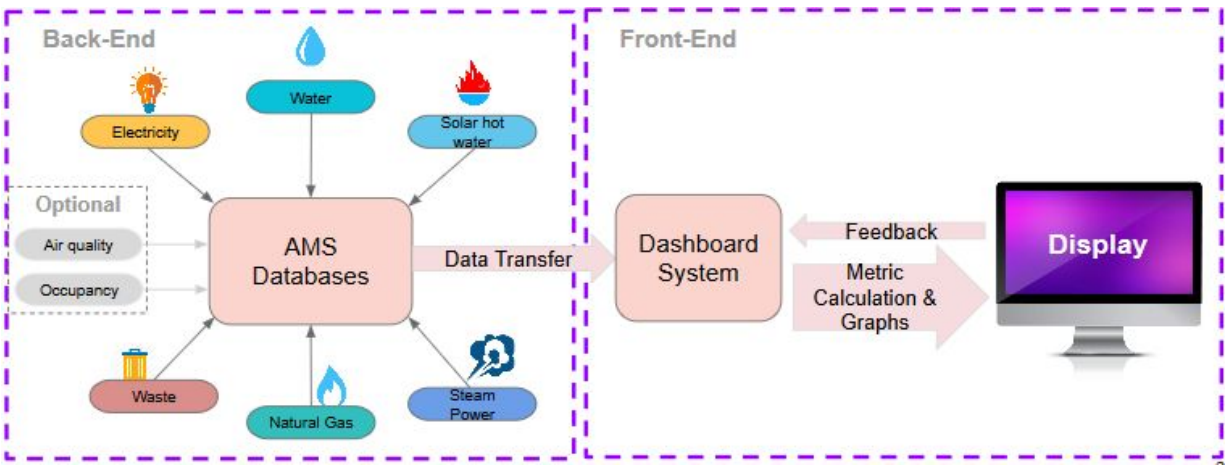


Illustration 1 - High Level Diagram

1.2 Requirements Traceability Matrix

The following two traceability matrices outline where each major component from the database and the dashboard were accomplished. This matrix is used to identify which requirements need further attention and additional requirements that may not have been addressed in future iterations.

Requirements	**Database Components**		
	Script for AMS Sensor Data transfer	Script for Waste Scale Data transfer	Database Structure
Database must be on an accessible server			Prototype 1
Data must date back one year before being pulled to local computer			(Client is responsible for data backup after one full year)
Integrated database verifies information			Prototype 2
Layout allows modification for additional sensors			Final Deliverable
Database ensures atomicity			Prototype 1
Units labeled & consistent			Prototype 1
Script does not crash when performing daily routine tasks	Final Deliverable	Final Deliverable	
Handles situation where Waste Computer, Sensor Database or Cloud Server is down	Final Deliverable	Prototype 2	

Requirements	**Dashboard Components**		
	Web Platform (visual studio)	Injunctive / Normative Slideshow	Graphs/Lists using integrated data
Run on a windows environment	Prototype 2		
Affect student behavior using injunctive feedback		Prototype 2	
Provide graphs and/or lists for research			Final Deliverable

2 System Architectural Design

2.1 Chosen System Architecture

We chose a windows operating system which would allow us to easily query data from the existing AMS Sensor Database, and the Microsoft SQL language would mean the languages between the systems are consistent.

2.2 Discussion of Alternative Designs

Two types of operating systems were considered, along with two types of software to carry out the design. The windows operating system is widely used within UBC, however there are additional costs for multiple users. Linux is fast and doesn't have license fees; however, it may be unfamiliar for future projects, and difficult to transition to a windows environment. An SQL database is relational and makes better correlations for visualizing data, however it requires more time to create the database and integrate data appropriately. NoSQL does not require predefined implementations of the data, and can easily include in-structured data-types, however future relationships cannot be built on top, making it difficult to build off of the database in the future.

2.3 System Interface Description

The database is structured such that the individual data, the units (i.e. Kg), and the waste type (i.e. recycling, garbage, paper) are separated into three tables so that additional sensors can be added without modifying the table layout. This satisfies our requirement because it is maintainable, and easily portable.

3 Detailed Description of Components

3.1 Back End Design

The Canadian Cloud server stores two SQL Server Express databases: sensor database and waste scale database. We will name these databases Server Sensor Database and Server Waste Database further in the document to avoid confusion with AMS Sensor Database and AMS Waste Computer. Two python scripts are responsible for data processing and data transfer. One of the scripts obtains data from the AMS Sensor Database, processes the data, then stores it in the Server Sensor Database. The other script obtains data from the AMS Waste Computer and stores it in the Server Sensor Database. Each script records errors during this process in a log file.

3.1.1 Server Sensor Database

This database stores resource consumption measured everyday by 3 types of sensors: electricity, natural gas, and water sensors. Each data entry represents the total resource consumption measured by a single sensor. Currently, we are monitoring readings from 15 electricity sensors, 2 natural gas sensors, and 1 water sensor. There is, however, opportunities for adding more sensors to the database if further research and design requirements change.

The data stored on the AMS Sensor Database for the sensors we are monitoring have two different forms of representation: readings that represent cumulative resource consumption and readings that represent the rate of resource consumption at a certain time. More specifically, the system stores cumulative resource consumption for the electricity sensors, and rate of resource consumption for natural gas and water sensors.

A python script runs on the Cloud Server, it is scheduled to perform its job everyday at 7.00am. When it is active, it first accesses the Server Sensor Database to retrieve a list of sensors being monitored. Then, for each sensor, it checks for the timestamp of the last data entry in the Server Sensor Database and stores it at Timestamp A. From this point, the script performs differently according to the type of sensor being processed.

For cumulative resource consumption readings:

> The script adds 1 day to Timestamp A to get Timestamp B, and then retrieves the reading of the sensor at Timestamp B and Timestamp A respectively. The difference between the two readings is then stored in the Server Sensor Database with Timestamp B.

For rate of resource consumption readings:

The script gets Timestamp B in the way, and retrieves all the data entries between Timestamp A and Timestamp B. It then calculates the time intervals between adjacent readings and multiplies them by the rates to get the resource consumption between every time interval. Finally, these values are summed up and added as a new entry into the Server Sensor Database with Timestamp B.

During the process described above, the script analyzes whether the electricity entries or any of the natural gas sensor rate values are negative. It was noticed that these two sensors contain some inconsistencies in their measurements. If total electricity consumption for one day is negative, it may be that the sensor is misconfigured or indicate an error from the AMS Sensor Database. If there are negative values present for natural gas, then they are often very small (< 1 cubic meter/per hour) and may be caused by the natural gas backflow in the pipes.

We cannot afford to display negative power useage or negative natural gas values on our dashboard since it makes data either unrealistic or uncomparable to different timestamps. If the script notices the negative electricity consumption on a day, it would replace the value with a “zero” in the server sensor database. If the script identifies negative rate measurements for natural gas it ignores those time intervals where negative measurements occur and sums up only intervals having positive measurements.

Negative values for water sensors have not been recorded yet, most likely because we only have 1 water sensor recording data. However, if any of the newly added water meters contain negative entries, they would be replaced with zeros as well.

The server sensor database schema is displayed below. Table SensorValue stores entries in the following fields: primary ID - the unique number of the table entry - entry date and time, sensor ID and value. SensorID is an integer number that is bound with the industrial sensor name - SensorName, and SensorTag - the common and more colloquial name of the sensor. UnitID field is an integer that refers to the Unit table binding ID to the real name of the unit.

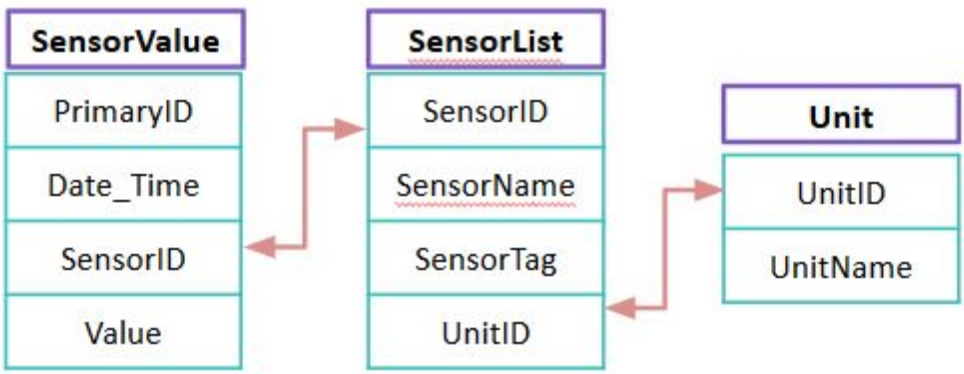


Illustration 2 - Server Sensor Database Diagram

The above implementation prevents frequent and repetitive use of sensor and unit names by replacing them with integers in the main table SensorValue. It saves space and allows additional sensors and units to be added separately.

For a better understanding of the system design, screenshots of the actual tables with data are included. These tables: SensorValue, SensorList, and Unit are shown in illustration 3.

PrimaryID	Data_Time	SensorID	Value
499	2015-06-12 19:00:00.000	8	0
500	2015-06-12 19:00:00.000	9	1750
501	2015-06-12 19:00:00.000	10	250
502	2015-06-12 19:00:00.000	11	2890
503	2015-06-12 19:00:00.000	12	130
504	2015-06-12 19:00:00.000	13	0
505	2015-06-12 19:00:00.000	14	0
506	2015-09-08 01:24:21.000	15	439.7...
507	2015-06-11 16:16:13.000	16	123.5...
508	2016-03-23 14:00:00.000	18	280
509	2016-03-23 14:00:00.000	19	0
510	2016-03-23 14:00:00.000	20	20

SensorID	SensorName	SensorTag	UnitID
3	JCL-NAE40/FCB-01.FEC-04.FM-04	FM-04	3
14	JCL-NAE43/FCB-01.VENDOR-72.Energy Total	Energy-18	1
15	JCL-NAE43/FCB-01.FEC-04.GM-02	GM-02	5
16	JCL-NAE43/FCB-01.FEC-07.GM-03	GM-02	5

UnitID	UnitName
1	kilowatt-hours
2	liters-per-second
3	liters
4	cubic-meters-per-hour
5	cubic-meters

Illustration 3 - Left: SensorValue Table, Top Right: SensorList Table, Bottom Right: Unit Table

3.1.2 Server Waste Database

This database stores weight measurements measured on a waste scale system, with multiple entries in each category of waste for each day.

Data stored on the Waste Scale Computer are stored as text files. AMS workers take garbage bins to the bottom floor of the Nest where they get weighed. Readings are processed and stored in text files by a java program. Entries in these text files are formatted in four columns:

timestamp, weight of waste, units, and waste type.

A python script runs on the Waste Scale Computer, it is scheduled to perform its job every hour. When it is active, it first scans for new data files in the directory where new data is saved. If new data files are found, they will be translated into query commands and executed by the script with new data inserted into the Server Waste Database. Processed text files will be moved to a subdirectory so that they will be eliminated from the next file scan. Illustration 4 displays the diagram of the Server Waste Database.

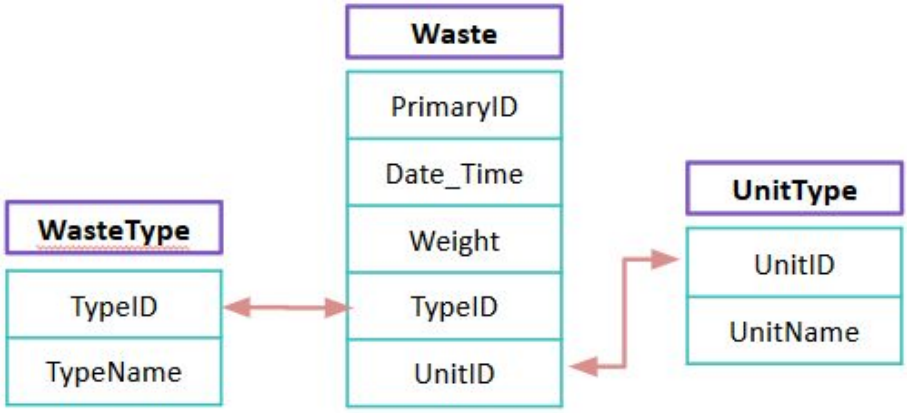


Illustration 4 - Server Waste Database Diagram

Table Waste, containing main data entries is connected to WasteType and UnitType tables that serve to substitute waste type names and unit names with integers in the main table, as mentioned earlier, to save space. In illustration 5 we will introduce screenshots of real table data to present its structure.

PrimaryID	Date_Time	Weight	TypeID	UnitID
46512	2014-06-19 12:02:01.000	61.8	1	1
46513	2014-06-19 12:04:59.000	62.8	3	1
46514	2014-06-19 12:05:45.000	63.5	4	1
46515	2014-06-19 12:07:33.000	68.1	2	1
46516	2014-06-19 12:07:44.000	1.8	1	1
46517	2015-04-30 12:32:52.000	59.8	1	1
46518	2015-04-30 12:35:36.000	61.5	3	1
46519	2015-04-30 12:45:51.000	10.6	1	1
46520	2015-08-05 12:54:00.000	54	3	1
46521	2015-08-05 12:55:20.000	54	2	1
46522	2015-08-05 12:56:08.000	54	4	1
46523	2015-08-05 12:57:04.000	54	1	1

	TypeID	TypeName
1	1	Recyclables
2	2	Paper
3	3	Organics
4	4	Others

	UnitID	UnitName
1	1	kg
2	2	g
3	3	lb

Illustration 5 - Left: Waste Table, Top Right: WasteType Table, Bottom Right: UnitType Table

4 User Interface Design

4.1 Description of the User Interface

The dashboard was required to, first, display metrics that would affect student behaviour, and second, display graphs and/or lists that would be available to students for research. Both of these also had constraints in regards to the consistency of colour choices, images, and how the data was displayed. Given that UBC building operations had a goal to, "...divert 60% of our operational waste from the landfill by 2016, and 80% by 2020," it meant that our project had an opportunity to address this issue, something that AMS Sustainability was also concerned about, in order to satisfy our requirement to affect student behaviour. The breakdown has a slideshow at the top of the web page, defaulting to behavioural feedback for when students walk by, and then additional tabs relating to the various sensors in the building. Each tab has a graph with a scaleable time-frame from one week to six months.

4.1.1 Screen Images

The following illustrates two methods used to satisfy our requirements by displaying user behaviour (illustration 6) and building metrics for research (illustration 8). The first is presented as a slideshow, implemented on the home page. The slides contain one character such as a water bottle with either a:

1. Tip to stay sustainable (Illustration 7), or
2. Feedback indicating how well, in this example, the user has managed to decrease waste (by increasing recycling).

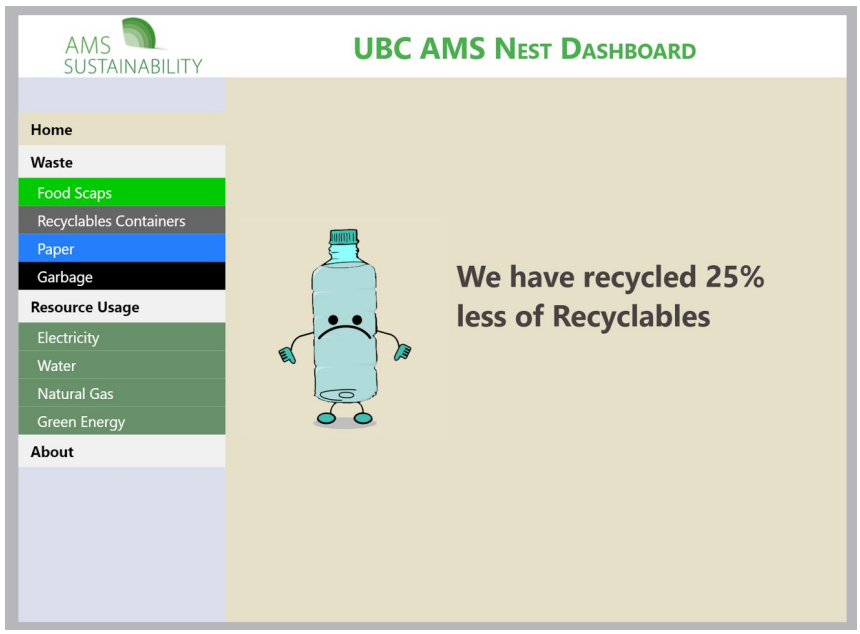


Illustration 6 - Dashboard display showing percentage-based feedback

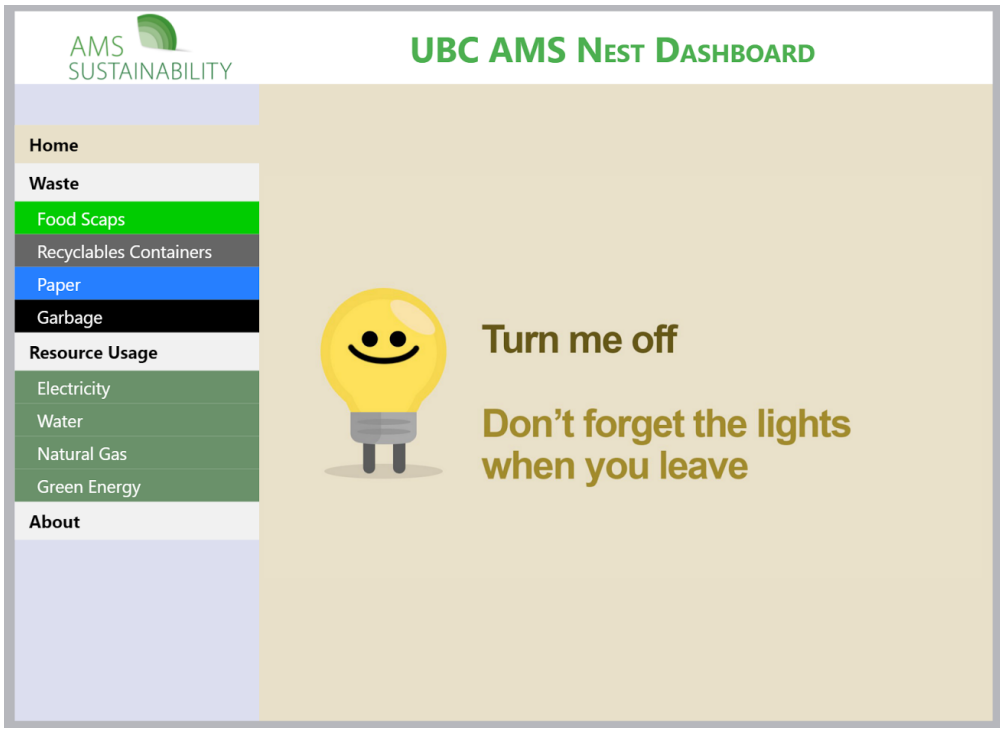


Illustration 7 - Dashboard display showing sustainability tip

In regards to the researchable data for students, we established a connection between the dashboard and our database to display the data with an interactive graph. This not only allows the user to observe real time data, but also makes research for long term behaviour possible. A screenshot of the dashboard is shown in Illustration 8.

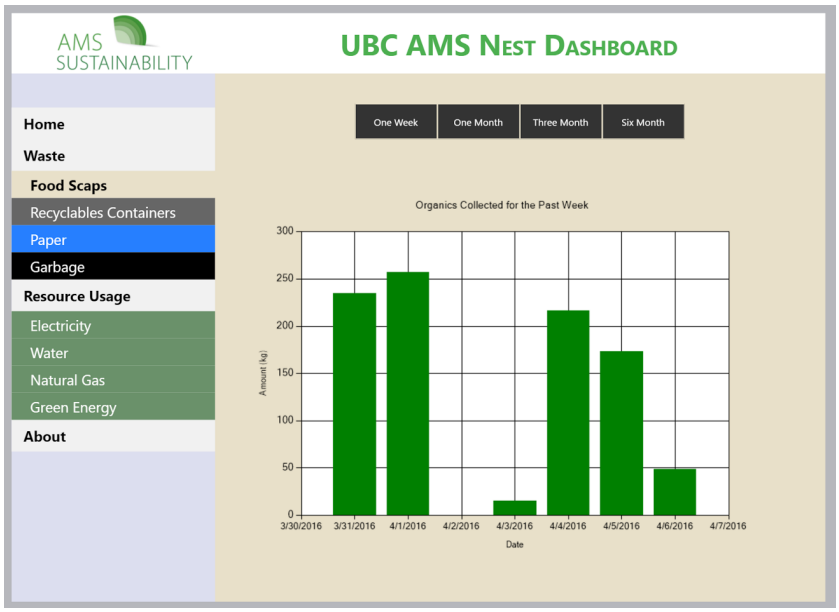


Illustration 8 - Dashboard display showing bar-graph of statistics

4.1.2 Objects and Actions

Similar to other easy to use websites, the navigation bar is placed on the left. The white tabs are the main categories, with coloured tabs as sub-menu's. Furthermore, the waste sub-menu are color coded to match the color schemes of the waste bins at UBC, as shown in Illustration 9, so the user will be able to quickly recognize the consistency in data types.

Illustration 9 - How the dashboard's display colors correspond to existing waste bins

The second half of the webpage contains graphs which displays trends within the AMS Sensor Database , as shown in illustration 10. The user is able to change the data range of the graph from one week to 6 months. After the user selects a range, the graph will load real time data as shown below.

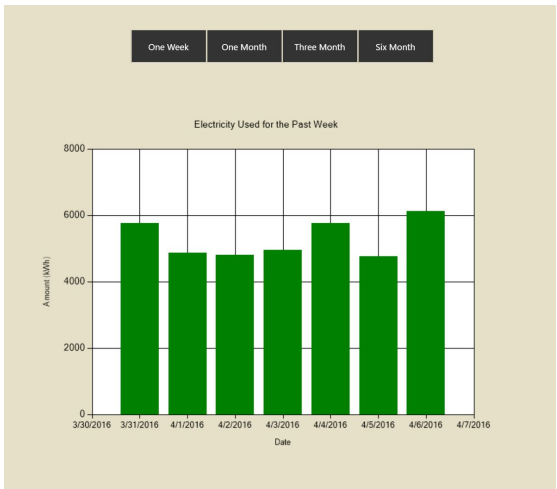


Illustration 10 - Dashboard bar-graph with time range sub-tabs

Illustration 11 shows the final deliverable for this project as a high-level flowchart with component parts.

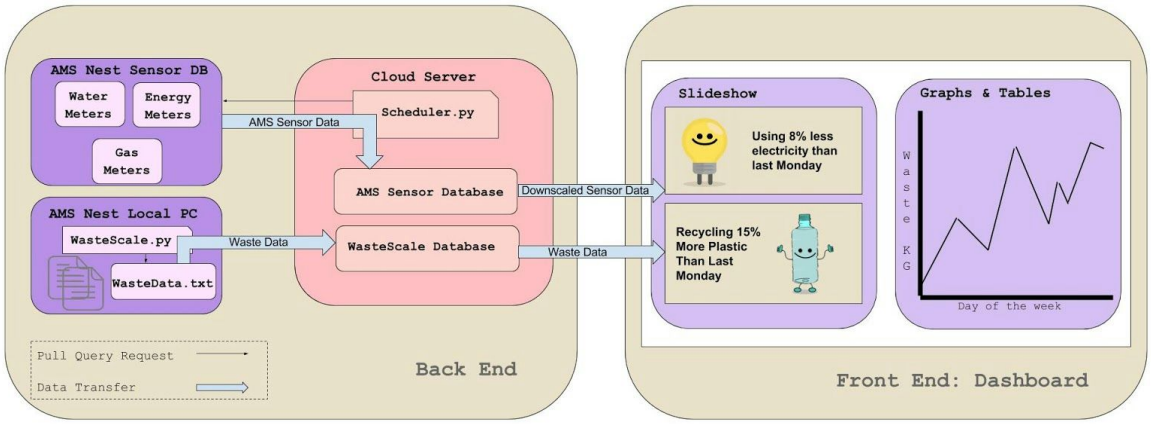


Illustration 11 - How the different components of this project relate to each other

The University of British Columbia

Verification & Validation Report

Integrating Various Building Metrics from the AMS Student Nest into a Presentable Dashboard (Campus + Community Planning, AMS Sustainability)

Prepared By:

Anna Gudimova

Euan Chow

Meng-Yeng (Dan) Lee

Micheal Griffin

Yuxin (Eugene) Xu

UBC AMS Sustainability

April 8, 2016

The following is a list of actions done by the designers of this project. These tests, checks, measurements or inspections are explicitly related to items in the Requirements specification.

Table of Contents

1 Introduction

1.1 System Overview

First, we introduce the top level design and schema in this section. Illustration 1 displays a top-level schema of the design.

The system consists of two main components: back end and front end. Back end resides on the Canadian Cloud server and contains SQL Server Express 2014 databases and data processing components - scripts written in Python 2.7. Front end, or the Dashboard, is a display that provides a user-friendly interface for analyzing and displaying data stored on the back end.

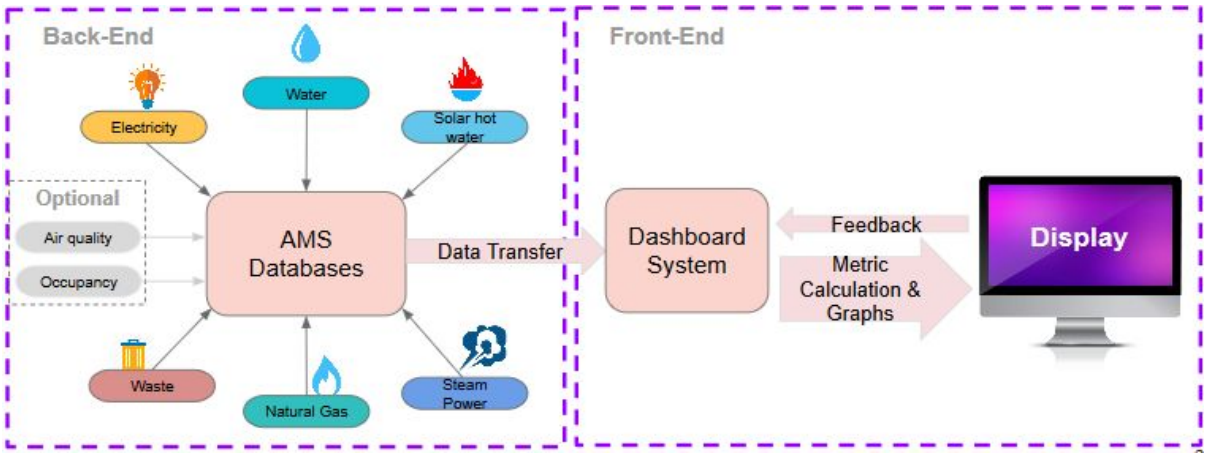


Illustration 1 - Top-level Design Schema

1.1.1 Back End

Back end integrates AMS sensor data for water, gas, and electricity measurements as well as AMS Waste Scale data for the amount of paper, recyclables, organic, and mixed garbage into two SQL databases stored on a Canadian cloud server. Sensor data for each sensor is pulled from the mirrored backup of the AMS Nest sensor database, downscaled to one entry per day per sensor and inserted into the Sensor database on the cloud server. AMS Nest waste data is retrieved from text files stored on a local AMS Nest computer and inserted into the Waste database on the cloud server.

1.1.2 Front End

On the front end C# application queries databases stored on the server, performs light data processing (like addition and comparison) and displays the data on our webpage for the dashboard. This screen is located on the bottom floor of the NEST next to the AMS Sustainability center.

1.2 Test Approach

In this section we will describe test approaches that we used for our system. We will also mention some risks that we foresee occurring throughout the lifetime of the system. Actual testing and risk mitigation strategies will be described in the following sections in more detail.

1.2.1 Back End

1. We want to test atomicity when attempting a query.
    a. I.e. what happens to the data being inserted if the connection is suddenly shut down or some other interruption happens. Will information be inserted with data corrupted/partial data or will the query not be executed at all.
2. We can predict that at some point in time either our Canadian cloud server or AMS Nest local computer may shut down or restart. Therefore, we need to make sure our scripts run after both computers reboot.
3. We have to test the correctness of both Server Sensor Database script and script for Waste Scale Database stored on the AMS local computer,
    a. I.e. we need to see that queries insert correct data and report errors in the log files properly.
4. We have to ensure data inconsistencies from the original mirrored AMS sensor database are not transferred to our database.
    a. Negative values, inconsistent measurement intervals, “dates from the future”, i.e. date values exceeding current date.

1.2.2 Front End

1. We want to make sure the Dashboard doesn’t crash if it gets no information from the databases. In that case, the Dashboard only displays informative slides,
    a. I.e. slides containing general recommendations for energy saving and no actual data.
2. If dashboard receives invalid (negative) values, even after we took every effort to eliminate these while integrating data (it still may happen as a result of later modifications) we need to make sure the invalid data is not displayed and only informative slides appear on the screen until the issue is fixed.
3. There is a possible risk of the dashboard losing Internet connection or crashing. This, however, is presumed to be immediately noticeable and we will address the risk associated with this later.

2 System Testing

The main idea for performing tests is to confirm that both back end and front end have expected and correct functionality. Thus, we assume that tests can help us find and eliminate program errors, however certain risks are inevitable, even though we layout the best approach for mitigation.

2.1 Features to be Tested: Back End

2.1.1 Atomicity Test

There is a risk that the scripts responsible for data transfer get interrupted by an unexpected event, e.g. computer power down, internet connect down, etc. MS SQL database feature atomicity, which is a property that ensures operations in our database either happen all at once, or nothing occurs. It prevents updates to the database occurring only partially when operations are interrupted. To confirm that atomicity is enforced on databases on the Canadian Cloud server, we ran long query commands that insert and delete multiple entries from the database. As the query commands are being processed, we terminated the process for the query execution and confirmed that no data had been inserted or deleted, preserving the atomicity of the system.

2.1.2 Server/Computer Shut Down

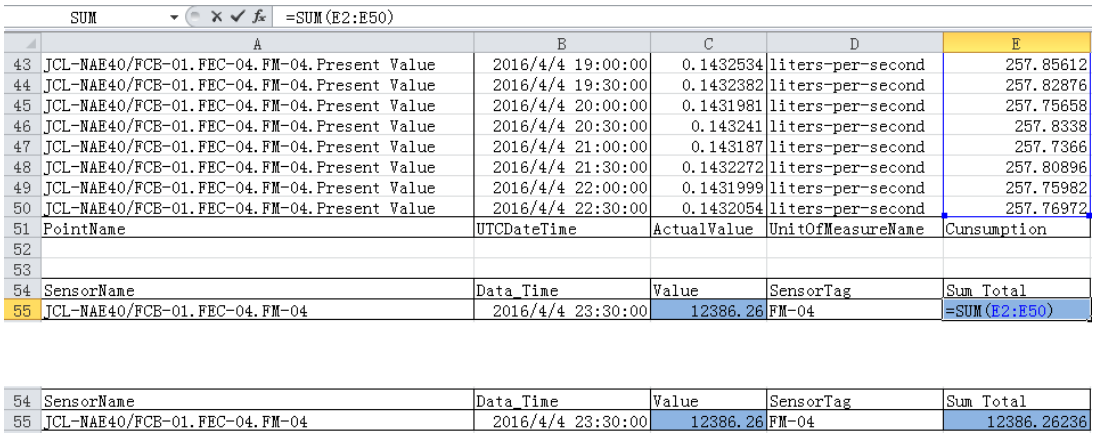
There is a risk that the Canadian Cloud server or AMS Nest local computer containing text files with waste entries may shut down or reboot. If this happens, we must ensure that our scripts run again and update the databases on the server for the time computers were down. In order to guarantee that the scripts run automatically after reboot we put the shortcut of the batch file (that runs the script) in the Windows Startup folder and check whether it runs after we restart the server. Script pulling information from the AMS sensor database automatically retrieves data for the number of days passed since the server went down. The Waste Scale script automatically translates text files into insert instructions if there is any data files waiting to be sent.

2.1.3 Data Correctness Test

The data transferred from the AMS Sensor Database to the Canadian Cloud server gets processed before they are inserted. The following ensures the calculation process is done correctly.

First, we randomly selected multiple data entries from the Canadian Cloud server, and compared them with the raw data stored on the AMS Sensor Database using Excel. If

results calculated by Excel matches, then the correctness of data is confirmed. The following illustration is a screenshot of the Excel comparing process.



SUM =SUM(E2:E50)

	A	B	C	D	E
43	JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016/4/4 19:00:00	0.1432534	liters-per-second	257.85612
44	JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016/4/4 19:30:00	0.1432382	liters-per-second	257.82876
45	JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016/4/4 20:00:00	0.1431981	liters-per-second	257.75658
46	JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016/4/4 20:30:00	0.143241	liters-per-second	257.8338
47	JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016/4/4 21:00:00	0.143187	liters-per-second	257.7366
48	JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016/4/4 21:30:00	0.1432272	liters-per-second	257.80896
49	JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016/4/4 22:00:00	0.1431999	liters-per-second	257.75982
50	JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016/4/4 22:30:00	0.1432054	liters-per-second	257.76972
51	PointName	UTCDateTime	ActualValue	UnitOfMeasureName	Cunsumption
52					
53					
54	SensorName	Data_Time	Value	SensorTag	Sum Total
55	JCL-NAE40/FCB-01.FEC-04.FM-04	2016/4/4 23:30:00	12386.26	FM-04	=SUM(E2:E50)

54	SensorName	Data_Time	Value	SensorTag	Sum Total
55	JCL-NAE40/FCB-01.FEC-04.FM-04	2016/4/4 23:30:00	12386.26	FM-04	12386.26236

Illustration 2 - Comparing data using Excel

2.1.4 Identifying Inconsistencies and Dealing with Them

Main data inconsistencies that threaten data integrity are negative data entries and unrealistic data that may appear as a result of sensor misconfiguration. Illustration 3 demonstrates what data entries stored in the AMS sensor mirrored database may look like. For gas meters - negative values caused by gas backflow; energy meters with sudden cumulative energy value drop by almost half leaving the possibility for large negative energy use for that day; and water meters with measurements “taken” far in the future - in 2100.

PointName	UTCDateTime	ActualValue	UnitOfMeasureName
JCL-NAE43:JCL-NAE43/FCB-01.FEC-04.GM-02.Present Value	2015-05-21 09:40:00.000	-0.4874005	cubic-meters-per-hour
JCL-NAE43:JCL-NAE43/FCB-01.FEC-04.GM-02.Present Value	2015-05-21 09:50:00.000	-0.5304456	cubic-meters-per-hour
JCL-NAE43:JCL-NAE43/FCB-01.FEC-04.GM-02.Present Value	2015-05-21 10:00:00.000	-0.5035967	cubic-meters-per-hour
JCL-NAE43:JCL-NAE43/FCB-01.FEC-04.GM-02.Present Value	2015-05-21 10:10:00.000	88.15934	cubic-meters-per-hour

PointName	UTCDateTime	ActualValue	UnitOfMeasureName
BIS-APPADX-PRD:JCL-NAE43/FCB-01.VENDOR-16.Energy Total.Present Value	2015-11-19 17:30:00.000	267110	kilowatt-hours
BIS-APPADX-PRD:JCL-NAE43/FCB-01.VENDOR-16.Energy Total.Present Value	2015-11-19 18:00:00.000	267120	kilowatt-hours
BIS-APPADX-PRD:JCL-NAE43/FCB-01.VENDOR-16.Energy Total.Present Value	2015-11-19 18:30:00.000	139770	kilowatt-hours
BIS-APPADX-PRD:JCL-NAE43/FCB-01.VENDOR-16.Energy Total.Present Value	2015-11-19 19:00:00.000	139770	kilowatt-hours

PointName	UTCDateTime	ActualValue	UnitOfMeasureName
BIS-APPADX-PRD:JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016-04-07 03:30:00.000	0.1434274	liters-per-second
BIS-APPADX-PRD:JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016-04-07 04:00:00.000	0.143516	liters-per-second
BIS-APPADX-PRD:JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2016-04-07 04:30:00.000	0.1434323	liters-per-second
BIS-APPADX-PRD:JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2100-03-06 09:00:00.000	1.671767	liters-per-second
BIS-APPADX-PRD:JCL-NAE40/FCB-01.FEC-04.FM-04.Present Value	2100-03-06 09:30:00.000	1.71802	liters-per-second

Illustration 3 - Top: Gas Meter Negative Values, Middle: Sudden Energy Jump, Bottom: Water Measurement Dated by 2100 Year

Our strategy is to turn all the negative values to “zero” for energy measurements, ignore intervals with negative rate when calculating the total gas consumption for the day, and retrieve only those dates that are smaller than the current date to avoid “measurements from the future” for water sensors.

2.2 Features Tested: Front End

2.2.1 Dashboard Receives no Data from Databases

When the dashboard is not presented with any information, the graphs will not load, and there will be no injunctive/normative slides indicating student behaviour in the Nest. Although this may seem like a problem, this satisfies our requirement to be atomic by avoiding stagnation of the display, showing old data irrelevant to students.

As an example, if for one week the dashboard was unable to connect to the database, this might mean that the dashboard would be displaying how well students saved energy from the previous month, when in reality, this data is one week old. Our system will prevent any information from being displayed until a live connection has been made, preventing this issue.

2.2.2 Dashboard Receives Invalid Data from the Databases

Should the dashboard be given negative values from the database, it will register this as an invalid data type, and prevent any injunctive feedback (like how much energy we saved) from being displayed. This too prevents users from receiving invalid information and being misinformed by corrupt data.

Instead of preventing any slides from showing, the slideshow is filled with two different images. Some that show general sustainability information like, “Unplug your electronics”, and others that give specific feedback using the database, “You have saved 5% of your electricity usage since last Monday”. The dashboard simply prohibits any slides with inaccurate injunctive information from being displayed, and only displays the slides with general information.

2.2.3 Dashboard Crashes or Webpage Loses Connection

Lastly, when the dashboard loses connection to the internet, or it crashes itself, the system becomes unresponsive, which is a potential risk especially if nobody is interacting with the display. If the users of the Nest often interact with the dashboard, then it is likely that someone will indicate that the system is not working.

3 System Diagnostics and Modification

In this section we describe how the end user of the system should interpret messages they see in log files or system behavior in general and give advice on how to address the issues that may arise during the system use. We also give recommendations on how user may modify the system to add new sensor type to it or new graph to the display.

3.1 System Diagnostics: Back End

UBC students, our target audience for the system, may not directly interact with the back end data stored on the databases, and it may not be immediately obvious to them that something has gone wrong. The only way for students to identify malfunctioning is by noticing that behavior on the front end has changed, i.e. dashboard displays either nothing or only a subset of images it should display. Another group of the target user is our client, the AMS Sustainability officers, who have access to the the Remote Desktop of both the Canadian Cloud server and the Waste Scale computer, and are able to check the log files. Shortcuts for log files are created on the desktop of both devices.

3.1.1 Newest record in log file is an Error message

On the Canadian Cloud server, if the newest log entry is an error, it means that the python script cannot connect to the AMS Sensor Database. Possible causes are:

Canadian Cloud Server is not connected to the Internet

User should right click on the Network Connection Icon on taskbar and choose the Troubleshoot Problems option. If the problem remains, reboot the server. If the problem remains after the reboot, user should contact Canadian Cloud client service to resolve the issue.

AMS Sensor Server is Down or Connection is Lost

User should contact UBC IT to resolve the issue.

On the Waste Scale computer, if the newest log entry is an errors, it means that the python script cannot connect to the Canadian Cloud server. Possible causes are:

Waste Scale Computer is not connected to the Internet

User should right click on the Network Connection Icon on taskbar and choose the Troubleshoot Problems option. If the problem remains, reboot the computer. If the problem remains after the reboot, user should contact UBC IT to resolve the issue.

Canadian Cloud Server is Down or Connection is Lost

User should check if the Cloud server is booted by logging in to the Canadian Cloud account page at https://www.cacloud.com/panel/index.php?/clientarea/ and go to the Services tab, then click on Vancouver SSD. If the status column does not display “Active”, go to the device setting page by clicking on the Server name and turn on the Server. If the status column displays “Active”, use remote desktop to access the Cloud server to perform a reboot.

3.2 System Diagnostics: Front End

We will analyze what may go wrong with the front end performance here and see what can be done to fix the issues that may occur.

3.2.1 Dashboard Displays Error Message

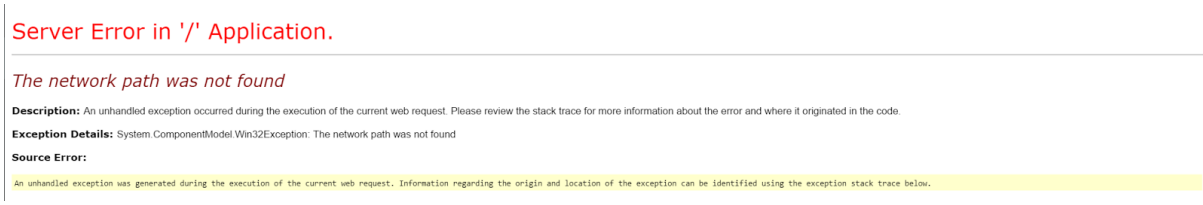


Illustration 4 - Error message on Dashboard

The Dashboard will display an error message if it is not able to connect to the Canadian Cloud Server. Possible causes are:

Dashboard Computer is not connected to the Internet

User should right click on the Network Connection Icon on taskbar and choose the Troubleshoot Problems option. If the problem remains, reboot the computer.

Canadian Cloud Server is Down or Connection is Lost

User should check if the Cloud server is booted by logging in to the Canadian Cloud account page at https://www.cacloud.com/panel/index.php?/clientarea/ and go to the Services tab, then click on Vancouver SSD. If the status column does not display “Active”, go to the device setting page by clicking on the Server name and turn on the Server. If the status column displays “Active”, use remote desktop to access the Cloud server to perform a reboot.

3.2.2 Dashboard Displays Only a Subset of Slides

There is a number of possible reasons for this behavior that we will list below:

Waste Scale Computer is Down or Connection is Lost

In this case messages about the waste consumption level will be missing on the screensaver. The user should check whether the AMS Nest local computer is down or the script is not running anymore.

If the computer is down please reboot it. Check if both the WasteScale software (A java program generating text files with waste data) and the script sending data to server waste database are running. If not, run them to activate data transfer process. Check the log files to make sure there has been no errors lately.

Mirror Database is not Accessible on the UBC Server or Connection to It Is Lost

In this case messages about energy, water, and gas consumption level will be missing on the dashboard screensaver. The user should check whether they can connect to the mirror AMS Sensor database manually (i.e., that credentials are still valid and TCP port is open for our server access). If not, contact project coordinators and UBC IT department to receive access back. If, however, the connection exists, check whether there are entries for energy, water, or gas sensors for the latest dates and whether they are valid, i.e. not negative and realistic. See the log files to make sure there has been no errors in the system lately.

Canadian Server is Down or Connection to It Is Lost

In this case neither sensor nor waste consumption level data will be displayed, only informative slide set with general recommendations for the public will be seen on the Dashboard screensaver. User should check if database credentials have been modified and try to connect to the cloud server manually. If they succeed the next step should be checking log files and whether data for the latest days exists at all.

Additionally, one general recommendation for debugging is to check if graphs with detailed information can be seen and if they contain entries for the last days.

3.3 System Modification

3.3.1 Back End Modification

When adding readings from a new type of sensor that is not in the AMS Sensor database.:

- User needs to create a new database on the Canadian Cloud server using either query commands or the GUI on Microsoft SQL Management Studio for new database creation. For consistencies, structure for the new

database should be similar to the server sensor database or the waste scale database.

- If the data source is a database
  - Write a query (SQL file) to calculate the total value per day measured by this sensor.
  - Write a script to run this query command regularly to update the data on the new database.

- If the data source is data files on a computer
  - Write a script to 1) check if new data files exist, 2) translate the datafiles into query commands, and 3) update the data on the new database by sending these query commands to the new database.

When adding readings from a new sensor from the AMS Sensor database to the server sensor database on the Canadian Cloud server, user should follow these steps:

- If the new readings are measured in units that do not exist in the Unit table, User needs to add new units in: tblUnit table
- Add the manufacturer's name of the sensor in: tblSensorList
- If this sensor is of a new type, i.e. not electricity, or natural gas, or water sensor, then user needs to write a query (SQL file) to calculate the resource consumption per day measured by this sensor.
- Modify the python script so that it uses a correct SQL file (either the SQL for electricity, or natural gas, or water sensor), or the new SQL file when processing readings from this sensor.
- Manually add a new entry to the server sensor database for this sensor with a timestamp, indicating the date of this sensor's earliest measurement on the AMS Sensor database.

Note: source code for the scripts are located in following directives.

- Canadian Cloud server
  - Script - C:\Users\Administrator\Desktop\CaCloud Script\Script\
  - Query - C:\Users\Administrator\Desktop\CaCloud Script\Query\
- Waste Scale computer
  - Query - C:\Users\student\Desktop\Capstone Waste Scale v1.0\

3.3.2 Front End Modification

When adding a new tab to the dashboard, user should follow these steps:

- Add an additional List item in Navigation class tag in every .aspx file along with the appropriate class depending whether if it's a sub-menu tab or a main category tab.
- Edit a preferred content inside the main class tag.
- Make sure to include the appropriate class to every other tags to have the same tab structure.

When adding additional graph controls, the user should follow these steps:

- Add an additional List item in ControlButton class tag with an additional TypeFive class tag

```
</div>
<div class="main">
    <div class="ControlButton">
        <ul class="ChartControl">
            <li class="TypeOne">One Week</li>
            <li class="TypeTwo">One Month</li>
            <li class="TypeThree">Three Month</li>
            <li class="TypeFour">Six Month</li>
        </ul>
        <div class="clear"></div>
    </div>
```

- Add TypeFive as a new function in the jQuery file so it will change graph content on click

```
$('.TypeFour').click(function () {
    $('.OneWeek').css('display', 'none');
    $('.OneMonth').css('display', 'none');
    $('.ThreeMonth').css('display', 'none');
    $('.SixMonth').css('display', 'block');
})
```

- Copy the whole SixMonth tag and paste it underneath SixMonth tag. Change the tag to an appropriate name. Make sure it is the same as the one in the jQuery file so it will respond.

```
<div class="ThreeMonth">
    <asp:Chart ID="Chart3" runat="server" BackColor="Transparent" DataSourceID="OrganicsThreeMonth" Palette="Bright" BorderlineDashStyle="Solid" Heig
        <Series>
            <asp:Series Name="Series1" XValueMember="DATE_DATE_TIME" YValueMembers="Sum_Weight"></asp:Series>
        </Series>
        <ChartAreas>
            <asp:ChartArea Name="ChartArea1" AlignmentStyle="All">
                <AxisY Title="Amount (kg)">
                </AxisY>
                <AxisX Title="Date">
                </AxisX>
            </asp:ChartArea>
        </ChartAreas>
        <Titles>
            <asp:Title Name="Title1" Text="Organics Collected for the Past 3 Months" BorderDashStyle="NotSet" Font="Microsoft Sans Serif, 10pt">
            </asp:Title>
        </Titles>
    </asp:Chart>
    <asp:SqlDataSource ID="OrganicsThreeMonth" runat="server" ConnectionString="<%$ ConnectionStrings:WasteScaleDBConnectionString1 %>" SelectCommand
</div>
```

- Edit the SQL query in the new chart tag, as well as the query string name and the graph title.

When adding additional slideshow from new data source, the users should follow these steps:

- Set the number of slide to show for.

```
int i = (int)ViewState["ImageDisplayed"];
if (i == 8)//number of slides to show in sequence
{
    ElectricalData(1);
    ViewState["ImageDisplayed"] = 1;
}
else
{
    i = i + 1;
    ElectricalData(i);
    ViewState["ImageDisplayed"] = i;
}
```

- Add a new string which contains the database name as well as a new string containing the sql querying the data.

```
string AMSSensorDB = null;
string WasteScaleDB = null;

SqlConnection connection;
SqlCommand command;
//Query Commands
string Elecsql = null;
string GEsql = null;
string NGsql = null;
string Watersql = null;
string FoodSsql = null;
string Othersql = null;
string Papersql = null;
string Recyclablessql = null;
```

- Copy and paste the first if statement
- Simply change *count* to the following number of the last case
- Change the datastring used to query the data

```
connection.Open();

//SensorDB
command = new SqlCommand(Elecsql, connection);
//query string loaded for connection

dataReader = command.ExecuteReader();
while (dataReader.Read())
{
    string result = Math.Abs(Math.Round(Convert.ToDouble(dataReader.GetValue(0)))).ToString();
    if (result == "")
    {
        return;
    }
    else
    {
```

- Determine the test cases and add new image to the folder directory

```
if (Math.Round(Convert.ToDouble(dataReader.GetValue(0))) >= 0)
{
    Label1.Text = "We have used " + result + "% more Electricity";
    Label1.Visible = true;
    Image2.ImageUrl = "~/Images/2.png";
    Image2.Width = 270;
    Image2.Visible = true;
}
else
{
    Label1.Text = "We have used " + result + "% less Electricity!";
    Label1.Visible = true;
    Image2.ImageUrl = "~/Images/1.png";
    Image2.Width = 270;
    Image2.Visible = true;
}
```

When the layout of the file needs to be configured, the user should follow these steps

- The style file is located in the styles folder
- Follow the comment in the file, find and section the user is looking to change

Project Continuation

The following is a list of suggestions for project continuation by our client

1. Install sensors which would record the number of people in the building
    a. Calculate average resource consumption per day per person.
2. Add more customization to the dashboard
    a. Users can select the type of graph to display the data
        i. Line graph - as an alternative to bar graphs for research
        ii. pie chart - to separate major categories of waste
    b. Customize data range
        i. E.g. from March to August
3. Add more interaction with the user:
    a. The users may be even more involved with the idea of waste sorting and energy
4. If users are given an opportunity to interact with the touchscreen display to see raw data and graphs, they would be given an opportunity to play a really simple game providing even more user behaviour and reinforcing sustainable habits in the Nest.
    a. This idea fits with another Capstone project where students are suggested to play a waste sorting game that helps them to remember how to sort waste.

Promotional Video

The video can be located at the following two addresses.
Capstone Youtube Account:

- https://www.youtube.com/watch?v=_6S3OHdeQZE&index=9&list=PLjFm8PBGO0F9tt3u-yeNG6A0G68s8YcHh

Micheal Griffin (Project Leader)

- https://www.youtube.com/watch?v=Zx698BrufCY

The video has also been delivered as a hard copy to our client.