Team ENGAGE

Preliminary Design

Easy Events (horrible name, ik (EasVent?))

**Leading Questions**

* What will our system provide that other data acquisition and management software’s/systems do not?
  + Ans: simplicity, efficiency, as much complexity as the SysAdmin desires
* How can we expand this to other clients/problems?
  + Event/Sensor model AND easy set-up will appeal to social scientists, politicians, professors. Basically anyone who wants to easily collect a refined set of data (ex student attendance/absence). Other systems are complex and can be used to handle complex, expansive data. These are not conducive to a professor trying to track only a few data points. Also, once one of these systems is set, they are difficult to change. Our system will be easy to configure and reconfigure. It will require no knowledge of code. Simple interface and set-up will attract non-technical and technical clients alike.
* How can we display data in simple, meaningful ways?
  + Ans: ??? See some answers below

**Definitions**

Event – An object with a uniform tag and set of parameters to track, gathered from sensors

Parameters – Set of data points associated with an Event, ex is time, GPS location, user, class,

Sensor – an installed instance of the app with the capability to accept plug-ins, send data to a master system via actual sensor readings, master requests (ex attendance ping), manual user input (ex absence request), etc.

Plug-in – A specific application of this engine. Will have a sensor and master side component. Managed by a SysAdmin. Has its own set of hardware permissions (ex location and time data). Data and permissions from one plug-in will not overlap with other plug-ins, so users won’t worry about being tracked or the app having access to data on a different plug-in.

SysAdmin– Owner(s) and creator(s) of a plug-in

Collection Method – How the data will be collected from the sensors. Two ideas currently are Ping and Self Response, which are both described below.

**Plug-in Set Up**

SysAdmin will access the completely GUI based set-up page. They are prompted to create an Event (tag and parameters) to track in their Sensors. They will also define the data parameters associated with an Event and the data type of said parameter. Parameter creation will be a tag and a drop-down of supported data types (time obj, string, int, GPS, etc.). They can also define the data collection method:

* Ping will be a message sent from the master to the sensors to request data
* Self Response will appear on the sensor app as an optional Event creation. For the attendance example, an Extension Request would be an option and they would fill in the specific fields.

Once they have their desired Events, they can choose to publish the Plug-in publicly (see **Future Ideas** section) or with a unique code/link which users/sensors can use to become sensors for that Plug-in.

**Sensor Set Up**

A user will install the app. They will be prompted to create an account as a unique sensor (will be important for applications like polling which rely on only one response per sensor). They can browse public plug-ins OR use the unique code to connect to a specific system, such as Larry Herman’s CMS250 class.

**SysAdmin Interface**

After the Plug-in is set up with desired Events to track, SysAdmin can choose to view list of Sensors (roster in attendance example) or a list of Events. For Event lists, each row is an event of the desired type (attendance events on 10/10) with columns of the parameter data. Will have lots of functionality to sort these tables/lists. I like the idea of color-coded bubbles since we were discussing graph implementations. Also, an Event should visually appear as a unique entity. As with everything in the app, the data visualization will be very configurable. We can also add functionality to run some basic stats and export the data as JSON, csv, Excel, etc.

**Attendance Plug-In Example**

Larry would create three Events for his plug-in. Each are listed here with their parameters and collection methods:

1. Attendance – Ping sensors every 5 minutes between 1-2pm every MWF
   1. User (String) – students unique UID
   2. Class (Int) – the class code (we are in cmsc 435)
   3. Time (time obj) – time when the sensor creates an Event in response to the Ping. Will show lateness!
   4. GPS Location (Location obj) – the sensors location at this time (are they in the correct place to prove they are attending class?)
   5. Status (Boolean(?)) – if the student is in the correct GPS location, their status will be TRUE. If they are not, and they have not submitted a validated (Status = TRUE) Absence Request, status will be FALSE, meaning they were not in class. All sensors will be pinged regardless of where the sensor is so we need to consider the Event created when they are not in a valid location at the time of ping.
2. Absence Request – Self reported by sensor when they think they have an excused absence
   1. User
   2. Class
   3. Time
   4. Reason (Int) – Larry can create a dropdown menu for this parameter where each dropdown item is one of the class defined excused absences (illness, death of a relative, class conflict, etc). It will remove ambiguity and discourage students from bullshitting an excuse.
   5. Proof (jpeg, PDF, docx, string, etc.) – a chance to upload a doctors note or some other form of verification. If the user selects “Other” in the Reason dropdown, they will have a textbox to write why they were absent.
   6. Status (Boolean) – one of the SysAdmin (TA or Larry) will have to verify the absence and change the status to TRUE if the reason and proof indicate a valid excused absence

**DNC Polling About Candidates Plug-In**

Another cool application of this engine is for social science or anonymous polling. A user could browse the public plug-in store and download this poll. It would contain just one Event which is the sensor’s response to the poll.

1. Poll Response – Self Reported
   1. Age
   2. Party Affiliation
   3. Voting State
   4. Registration Zipcode (to determine district)
   5. Question1 Response – again, Nancy Pelosi can create the dropdown menu with response options (probs put Warren as option A)
   6. Question2 Response
   7. …

This is a perfect example of a quick and easy way for a non-technical client to use our engine to gather data. They could easily create and disseminate the poll, get responses, and view the results.

**Future Ideas**

* Have a public store of plug-ins for any user with the app to become a sensor for other data projects (political polling, social science, etc.)