CS 1520 / CoE 1520: Programming Languages for Web Applications (Spring 2012)

Department of Computer Science, University of Pittsburgh

Term Project: Super-Advanced PIttsburgh Emergency Notification System (SAPIENS)

Released: April 15th, 2012 **Due:** 11:59pm, Thursday, April 26th, 2012

Goal

Build a realistic project utilizing web programming, while integrating with external APIs.

Description

You are asked to build SAPIENS, the Super-Advanced PIttsburgh Emergency Notification System, using PHP, MySQL, and Javascript/AJAX. SAPIENS is designed to overcome the one-size-fits-all approach to notifications, by providing a means to distinguish among:

- Different areas of interest, based on static subscriptions by users,
- Different areas of interest, based on dynamic user location (acquired by means of a virtual "check-in" to that location),
- Different levels of severity of the notifications (**1**: non-critical, **2**: critical, **3**: very-critical), and
- Different communication mediums (namely: web, email, and text messages).

The main idea is that users of SAPIENS register subscriptions for locations of interest (statically or dynamically) and get notified when an event is generated for locations they have subscribed to. Notification can happen by web (i.e., when they login to SAPIENS), by email, or by text message (implemented through email), according to the event's level of severity and the preferences of the user.

Database Schema

The database schema is provided at the following URL:
`http://db.cs.pitt.edu/courses/cs1520/spring2012/assign/a5.project.sql`
You should use this in order to create the set of tables needed for the project, in your own database.

The database consists of the following tables:

Users Table Users is used to record the profile information of every user, including his/her contact information (email and cell phone), as well as the location of his/her last check-in. Notice that you only need to record the last location, and this should overwrite any previous ones. Also, `user_id` is assigned by MySQL. Please note that `user_email` is the user's email address whereas `user_cell_email` is the email address that corresponds to his/her cell phone, in order to receive text messages. In order to simplify things, you should determine `user_cell_email` when the user is adding his/her information. In particular, you should use the `user_cell_provider` information, to get `cp_template` (e.g., @vtext.com for Verizon), which you will then combine with `user_cell_phone` (e.g., 4126248843) to create `user_cell_email` (e.g., 4126248843@vtext.com)[1].

Table: Cell_Providers

Column	Type	Null?	Default
cp_id	int(11)	No	
cp_name	varchar(25)	No	
cp_template	varchar(50)	No	

[1]Information for this mapping has been taken from http://www.makeuseof.com/tag/email-to-sms/

Table: Users

Column	Type	Null?	Default
user_id	int(11)	No	
user_firstname	varchar(20)	No	
user_lastname	varchar(30)	No	
user_login_name	varchar(20)	No	
user_login_pass	varchar(20)	No	
user_email	varchar(100)	No	
user_cell_phone	varchar(10)	No	
user_cell_provider	int(11)	No	
user_cell_email	varchar(100)	No	
last_login_ts	timestamp	Yes	NULL
last_loc_id	int(11)	Yes	NULL
last_loc_checkin_ts	timestamp	Yes	NULL

Locations Table Locations is used to record the different locations of interest, along with their relationships. In particular, we allow for a hierarchy of up to four levels (e.g., City of Pittsburgh $\leadsto$ University of Pittsburgh $\leadsto$ Sennott Square Building $\leadsto$ Department of Computer Science). The latitude and longitude fields are used for Google Maps, whereas the description field could be anything (e.g., the full street address).

Table: Locations

Column	Type	Null?	Default
loc_id	int(11)	No	
loc_name	varchar(40)	No	
loc_description	varchar(120)	No	
loc_latitude	double	No	
loc_longitude	double	No	
parent_loc_id	int(11)	Yes	NULL

Subscriptions Table Subscriptions is used to record the interests of different users, in terms of locations. Every subscription for a location by a user needs to also include information about how the user should be notified, according to the severity of the event. For example, `min_severity_web` = 1, means that for events with severity 1 and above, the user expects to see them on the web site; `min_severity_email` = 2, means that for events with severity 2 and above, the user expects to get an email notification; `min_severity_cell` = 3, means that for events with severity 3 and above, the user expects to get text message notifications.

Table: Subscriptions

Column	Type	Null?	Default
sub_id	int(11)	No	
user_id	int(11)	No	
loc_id	int(11)	No	
min_severity_web	enum('1', '2', '3')	No	1
min_severity_email	enum('1', '2', '3')	No	2
min_severity_text	enum('1', '2', '3')	No	3

Dynamic Subscriptions Table Dynamic_Subscriptions is used to store user preferences for the case where an event is matched against their current location (with the implicit assumption that one is always interested in events related to a location he/she is currently in). In particular, a user is considered being at a specific location, if he/she has checked into that location within the last 2 hours. The main preference here is which communication medium to use for the different severity levels (as was the case for the regular subscriptions).

Table: Dynamic_Subscriptions

Column	Type	Null?	Default
dyn_sub_id	int(11)	No	
user_id	int(11)	No	
min_severity_web	enum('1', '2', '3')	No	1
min_severity_email	enum('1', '2', '3')	No	2
min_severity_text	enum('1', '2', '3')	No	3

Events Table Events is used to record new events and their severity level, linked to a specific location.

Table: Events

Column	Type	Null?	Default
event_id	int(11)	No	
loc_id	int(11)	No	
event_severity	enum('1', '2', '3')	No	
event_ts	timestamp	No	CURRENT_TIMESTAMP
event_description	varchar(120)	No	

Event Matching

An event will match a user's subscriptions in the following two cases:

- the location of the event matches the location of a user's subscription (table *subscriptions* handles this).
- the location of the event matches a user's last checkin location and based on current time it has been under 2 hours since the user had checked in (table *dynamic_subscriptions* handles this).

According to the user's preferences, the matched event could simply be displayed when the user logins (i.e., via web), lead to an email message or lead to a text message notification via email[2]. Note that there is no additional implicit or explicit event matching (i.e., we ignore the hierarchy of the locations for matching purposes).

Web Pages

You are asked to implement the following web pages:

1. **sapiens.php** – this is the main entry page to your program, it should have a Login form (showing first) and a Register form (showing after one clicks Register). After successful login, the new page has links to the remaining pages (and the last login field is updated). After an unsuccessful login, a Register form is displayed.

[2]http://php.net/manual/en/function.mail.php

2. **event_list.php** – this is the page listing all events of interest to the particular user. There are two options: list only those since the last time he/she logged in (=default option) or list all notifications (relevant to the particular user), since the beginning.
3. **event_map.php** – same as the notify_list.php, but instead of a text list, you need to show the events on a map (using Google Maps API).
4. **subscriptions.php** – this page lists the user's current subscriptions and allows him/her to add new ones. When adding new subscriptions, the location is identified through a search box that supports autocompletion (using AJAX) from the list of all possible locations (using both loc_name and loc_description fields). Specifying the severity level is implemented as a drop-down menu.
5. **checkin.php** – this page allows a user to "check in" to a certain location (e.g., when going to class in a specific building). Selection of the location should be done in a hierarchical fashion, starting from the most general location (i.e., one where parent_loc_id = NULL) and work downwards. The user should be able to stop at any level (i.e., check-in at the University of Pittsburgh).
6. **new_event.php** – this page allows a user to add a new event to SAPIENS. The location is selected with autocomplete, as with the subscriptions.php page. Only users with subscriptions on a given location or having checked in a given location, can add new events for that location. Once an event is submitted, SAPIENS will detect whether any notifications need to be send out, and if yes, it will do so.

What to submit

Make sure that all paths are **relative**. Submit all files as a single zip file.

Academic Honesty

The work in this project is to be done *independently or in groups of 2 – 3 people*. Discussions with other students on the assignment should be limited to understanding the statement of the problem. **Cheating in any way, including giving your work to someone else, will result in an F for the course and a report to the appropriate University authority for further disciplinary action.**

How to submit your assignment

We will use a Web-based assignment submission interface. To submit your assignment:

- Go to the class web page `http://db.cs.pitt.edu/courses/cs1520/spring2012` and click the Submit button.
- Use your pittID as the username and the password you specified at the contact information form for authentication. There is a reminder service via email if you forgot your password. You must have already submitted your contact information, if you have not yet you need to do so now.
- Upload your assignment file to the appropriate assignment (from the drop-down list).
- Check (through the web interface) to verify what is the file size that has been uploaded and make sure it has been submitted in full. **It is your responsibility to make sure the assignment was properly submitted.**

You must submit your assignment before the due date (11:59pm, Thursday, April 26th, 2012) to avoid getting any late penalty. The timestamp of the electronic submission will determine if you have met the deadline. There will be no late submissions allowed after 11:59pm, Thursday, April 26th, 2012.

[Last updated on April 15, 2012 at 2:32pm EST]