CS 1520 / CoE 1520: Programming Languages for Web Applications (Spring 2013)

Department of Computer Science, University of Pittsburgh

Term Project: Pittsburgh Interactive Research Accounting System (piras)

Released: April 9th, 2013 **Due:** 11:59pm, Saturday, April 27th, 2013

Goal

Gain practical experience from building a real interactive web-database application, using PHP, AJAX, and MySQL.

Description

You are asked to build `piras`, the Pittsburgh Interactive Research Accounting System. The system is used to keep track of the finances of all research projects, most of which receive funding from external sources such as the National Science Foundation (NSF) and the National Institutes of Health (NIH).

The description and design of `piras` is based closely to the actual requirements for research accounting in the University of Pittsburgh. There are many simplifications however and you are only asked to implement part of the full functionality, in order to focus on the interactive features of the system.

Database Schema

You should create in your assigned MySQL database (using the script provided) and use in your program the following two tables:

Transactions that has the following fields:

- *txid*, a sequence number/ID for each transaction, that is automatically generated by MySQL every time you insert a tuple (you need to leave it NULL when performing an insert statement).
- *txdate*, the date the transaction was posted in `piras`.
- *account*, the account number that the transaction will be associated with. Note that the account naming scheme is complicated so this needs to be a string.
- *subcode*, the subcode that the transaction will be associated with. This refers to the type/category of this transaction (e.g., `6000` is for office supplies). The subcodes are the same for budget and for expense transactions.
- *amount*, the dollar amount of the transaction. A positive number indicates an *expense* (e.g., a purchase), whereas a negative number indicates an addition of money, i.e., *budget*, for the particular account and the particular subcode (i.e., part of the awarded budget for the grant).
- *description*, a textual description of what the transaction was.

Subcodes that has the following fields:

- *subcode*, a subcode number (e.g., `6000`).
- *description*, a textual description of what the specified subcode is used for (e.g., `Office Supplies`).

The Subcodes table is supposed to list all allowable subcodes. This is typically initialized at the beginning of your program.

Functionality

You should implement a PHP script, called `piras.php` that is the main entry point to your system. You should include your full name and pitt account name as part of this page and all other pages, as part of an "identity" banner on the top of the page. The `piras.php` script should display two statistics about the current database (the number of subcodes and the number of transactions that are currently in the database), and provide the following functionality:

1. **Reset DB**
   Link to `resetdb.php`, that initializes the database by deleting all transactions and all subcodes. However, the tables should **not** be dropped from the database.

2. **Import Subcodes**
   Button, next to a text input form that you should use to provide a URL that has a CSV file with subcodes. When the button is clicked, `import_subcodes.php` is called (using the GET method) to process the URL and add the data to the database. After execution, there should be a status line saying how many subcodes have been read and added to the database, plus a link back to `piras.php`. A sample CSV file will be provided.

3. **Import Transactions**
   Button, next to a text input form that you should use to provide a URL that has a CSV file with transactions. When the button is clicked, `import_transactions.php` is called (using the GET method) to process the URL and add the data to the database. After execution, there should be a status line saying how many transactions have been read and added to the database, plus a link back to `piras.php`. A sample CSV file will be provided.

4. **Maintenance**
   Link to `maintenance.php`, that allows a user to add new transactions, or delete/update existing transactions. This is expected to be very interactive and rely on AJAX. More information next.

5. **Querying**
   Link to `filter.php`, that allows a user to interactively filter existing transactions. This is expected to be very interactive and rely on AJAX. More information next.

maintenance.php

This page should list all transactions currently in the database, in reverse chronological order, i.e., most recent ones should be on the top of the page. Every transaction should be listed as a single row. There should be an "update" and a "delete" button at the front of every transaction. There should be an "Add new transaction" button at the top of the list. All of the functionality listed below needs to happen without the page reloading (i.e., you must use AJAX).

- **delete**: when a user clicks this button, there should be a popup confirming deletion. If the user clicks no, nothing happens. If the user clicks yes, then the transaction is removed from the database and is also removed from the displayed list.

- **update**: when a user clicks this button, all fields for the corresponding transaction (except for the *txid*) become text boxes with the current values in them, and a "Save" and a "Cancel" button are added at the end of the row. The user is allowed to modify any field and in the end click Save for the

changes to be reflected both in the currently displayed list and also in the database, or click Cancel, for the changes to be ignored and the previous version of the transaction to be displayed.

Please note that for the subcodes field, you need to create a drop down menu with only the allowable subcodes, that are currently in the Subcodes table. For the date field, the input format should be mm/dd/yyyy. These apply also when adding a transaction.

- **add new transaction**: when a user clicks this button, a row of empty text boxes are created at the top of the page (similarly to the update option, but without any initial values), and a "Save" and a "Cancel" button are added at the end of the row. There should not be any *txid*, this will be initialized by the database. After data is added, the user can either click Save or Cancel. Save means that the entry is added to the database, and also to the displayed list below (along with the new *txid*), whereas Cancel means any new transaction information is lost.

filter.php

This page should list all transactions currently in the database, in reverse chronological order, i.e., most recent ones should be on the top of the page. Every transaction should be listed as a single row. The top of the page should have a few options that allow a user to quickly filter the list of transactions that are being displayed. Note that this filtering should happen without the page being reloaded (i.e., you must use AJAX).

You should implement the following filter options:

- **keyword**: this should be a text box that will be used to only display transactions that contain the specified keyword as a substring in their description field, and eliminate all else from viewing. This can be reset by someone clearing the text box.

- **accounts**: this should be a series of checkboxes, one for each account number that exists in the Transactions table. By default all should be checked, but if one is unchecked by the user it means that transactions on this particular account should be removed from the listing.

- **subcodes**: this should be a series of checkboxes, one for each valid subcode number that exists in the Subcodes table. By default all should be checked, but if one is unchecked by the user it means that transactions on this particular subcode should be removed from the listing.

- **amount range**: this should be two text boxes, one for the minimum threshold value **min** and one for the maximum threshold value **max**. There are both empty by default. If either one is specified, then only transactions meeting this requirement (e.g., have amount $>$min, if min is specified) should be displayed. Note that one can provide only one of the two thresholds (i.e., just a min value, or just a max value). These fields can be reset by someone clearing the text boxes.

- **reset**: this should be a button or a link to reset all filter conditions to their default values, i.e., no keyword, all accounts, all subcodes, no amount range, and thus show all transactions. Note that you should not completely remove all transactions from your Javascript data structure when they are not supposed to be displayed, but rather make them invisible.

- **filter**: this button should be pressed by the user after he/she has modified the filter conditions (including clicking reset) in order for the filter conditions to be reevaluated and the list of transactions that are displayed to be refreshed.

Note that this is one part of your system that you are highly recommended to utilize a third-party library (e.g., one based on jQuery) to simplify managing the results (e.g., as a table).

BONUS

If you believe you need extra credit, you can consider the following additions to your program:

- **slider**: use a slider tool (from an external library) for the amount range, that you initialize with the lowest and highest values for the amounts and give the user a change to change the two endpoints, to filter some transactions out. Note that transactions can have both negative and positive amounts.

- **date tool**: use a popup calendar (that materializes when a user clicks within the text box) to supply date information in the "add a transaction" and "update a transaction" operations.

- **date range**: add another filtering option, based on the date of the transaction, which would only display transactions within a user-specified data range.

- **date range tool**: use the date tool as the input method for the data range filter operation (described above).

What to submit

Your program should be made to work with the provided private MySQL database that should be initialized to the schema provided. You do need to provide any files for the database initialization (i.e., must let the tables be in your database).

You should submit all the files needed for your program to work, along with your javascript libraries that should be in the same directory (i.e., using relative paths). There is no standard naming scheme for the libraries, although the expectation for javascript files is that they end in `.js`.

You should also submit a `README.txt` file that briefly explains the sources of any external libraries that you used in your code and also the feature they provided (for example date input popup page, slider, etc).

Academic Honesty

The work in this assignment is to be done *independently*, by you and only you. 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:

- If you have more than one file to submit, prepare your assignment for uploading, by generating a single zip file with all the files.
- Go to the class web page `http://db.cs.pitt.edu/courses/cs1520/spring2013` 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, Saturday, April 27th, 2013) 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, Saturday, April 27th, 2013.

[Last updated on April 10, 2013 at 3:21am EST]