

WBS Manual

Version 1.6

SK Knudson

May 23, 2013

Copyright ©2010-11 Stephen K Knudson. All rights reserved,

This document and the accompanying computer files are distributed in the hope that each will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Contents

1	Introduction	4
2	Purpose	4
	A scoring tool	4
	Scoring versus grading	5
	Side Effect	5
3	Web Based Scoring in Operation	6
	From the Point of View of an Individual Student	6
	From the Point of View of the Instructor	6
4	Why bother with computerized scoring?	7
	Motivation and Goals	7
5	Requirements	7
6	Installation	8
	Installation Overview	8
	In the box	9
7	Configuration: Data Submission Files	10
	Introduction	10
	<i>Editing</i> a database file	10
	Configure three files	11
	Appearance	16
8	Configuration: Scoring Student Data	16
	What can be scored?	18
	How to instruct WBS to score	19
	What can't be scored?	22
	Other Options	22
9	Technical Details	22
	Introduction	22
	Design Principles	23
	A quick primer on the use of php files	24
A	Screens for Data Submission	27
	Wbsindex.html	27
	Selections.php	28
	Gatherdata.php	29
	Storedata.php	30
B	Computer Requirements	32

1 Introduction

This manual provides complete instructions for the installation and use of **WBS**, a free web based tool for scoring student laboratory reports, supplementing the instructor's grading.

In next section, we discuss in more detail the purpose of **WBS**. The third section describes how **WBS** works, first from the point of view of the student and then from that of the instructor. Overall installation instructions are contained in the following section, with detailed descriptions of the various steps concluding the body of this manual.

Screens seen by students during the submission process are presented as figures in Appendix A, and a summary of computer and software requirements is presented in Appendix B. The requirements are minimal, as the software was developed with the intention that it be universally available, at least to educational institutions in the US and Canada.

2 Purpose

A scoring tool

WBS provides the laboratory instructor with a free tool offering rapid, accurate, and uniform scoring of student reported data, presumptively in a laboratory setting. For those skeptical about computer scoring, it may be important to emphasize that this may not be what you expect. **WBS** is not a fancy multiple choice machine. Rather, it should be regarded as a tool specialized for the science laboratory, where accurate measurement, calculation, and reporting are emphasized and therefore checked. As it is designed to minimize tedium and ensure accuracy, it also allows instructors to devote more time to instruction than marking routine aspects of student papers. While **WBS** is not intended to replace professional judgements, it does incorporate profession judgement via its implementation of your scoring key. With this approach, **WBS** is not intended to provide a grade, but rather to provide a score as one component of the grading.

The approach provides support for the instructor in a laboratory setting, regardless of the manner in which the course is presented, since all courses require students to make measurements, carry out calculations, and report results. Data-driven aspects of these stages can be scored.

The target placement for **WBS** is a lower division science laboratory with multiple sections and multiple teaching assistants. The approach addresses

- courses with large numbers of students. The instructor in courses with only a few students can carry out the tasks without the assistance of the product.
- grading uniformity across all sections. Students generally accept the notion that computer scoring is carried out in an unbiased manner, and that their interaction with

laboratory personnel has no influence on the score. And we are all taught to believe that the computer is accurate; in this case, it is as accurate as the information provided by the instructor. While we have found teaching assistants to be conscientious, and willing to follow a grading key, individual differences do turn up; this should not be a surprise.

In any event, **WBS** eliminates even the perception that the student would have been better off in a different section as far as the scoring is concerned.

- rapid return of scored laboratory reports to students. We think it important to provide this information to students before the day of the next laboratory experience, so that they can be informed if technique or calculations or any other similar aspect of the laboratory report needs to be considered. A well-written report on improperly handled data is fundamentally flawed.

Scoring versus grading

We have already attempted to make a distinction between scoring and grading a laboratory report. We mean the latter to be the assignment of a grade based on the student's work, regardless of the form of the submitted material. As such, the grade reflects the professional judgement of the instructor.

We mean the term scoring to refer to marking some of the mundane but important information gathered by the student. This takes special importance in the laboratory setting, where the student is expected to make and record measurements accurately, to carry out calculations based on those measurements, and to report information correctly. However, when hundreds of students are submitting weekly reports, the task of organizing and checking this material becomes large, especially since some of the data submitted can legitimately vary from student to student.

Side Effect

In the operation of **WBS** student data is collected in a database for the laboratory. This data may be examined for purposes other than scoring individual reports, in particular to determine if the experiment as implemented is meeting its pedagogical goals.

We are aware of only one other approach similar to this, WebMark[?]. It appears to be less general and less flexible than the current offering.

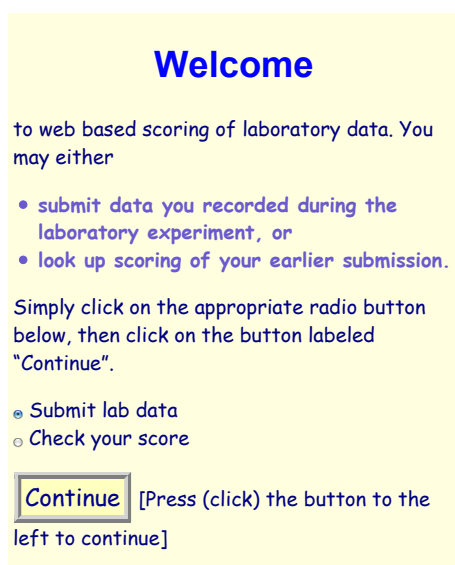
To understand what **WBS** does and does not do, we next present it in operation.

3 Web Based Scoring in Operation

In this section we examine **WBS** from an operational point of view. First we consider how each student interacts with **WBS** during or at the end of the laboratory experience. Then we consider how the instructor interacts following the laboratory.

From the Point of View of an Individual Student

At the completion of the experiment, the student uses a web browser to load a specified PHP file, `wbsindex.html`; the rendition is shown below and more legibly in Figure 2.

A screenshot of a web browser displaying a "Welcome" message. The title "Welcome" is in blue. The text says "to web based scoring of laboratory data. You may either" followed by two bullet points: "• submit data you recorded during the laboratory experiment, or" and "• look up scoring of your earlier submission." Below this, it says "Simply click on the appropriate radio button below, then click on the button labeled 'Continue'." There are two radio buttons: "Submit lab data" (selected) and "Check your score". At the bottom, there is a "Continue" button with a box around it, and a note "[Press (click) the button to the left to continue]".

Welcome

to web based scoring of laboratory data. You may either

- submit data you recorded during the laboratory experiment, or
- look up scoring of your earlier submission.

Simply click on the appropriate radio button below, then click on the button labeled "Continue".

☒ Submit lab data
☐ Check your score

Continue [Press (click) the button to the left to continue]

(The acronym "PHP" is discussed below; by default, such files are loaded by a web browser.) This file provides student access to **WBS** throughout the term, so it may for example be bookmarked. In response to prompts, the student logs in, specifies his/her section, and selects the appropriate experiment from a list. Next, and again in response to prompts, a student submits selected experimental data and results. This submission is echoed to the student for confirmation or correction; upon confirmation, the data (and name of the student) are stored in a database. The data submission process does not take students long. All of these screens are shown in Appendix A.

Several days later the student may, starting from that same file, use a browser to find out the scored results for the specified experiment. Thus, the student need bookmark only one (1) file for the duration of the course.

From the Point of View of the Instructor

To obtain scores, the instructor or designate simply loads a specified PHP file, the same file name for each experiment, in a web browser. This initiates the instructor version of the scoring program. Specification of the section and experiment returns an alphabetical list of students and scores for that section or, optionally, all sections.

A flowchart of program operation is given in Figure 1 on page 17.

4 Why bother with computerized scoring?

Motivation and Goals

We can only speak to what motivated us in the academic environment here. As seems to be common, we operate large enrollment chemistry labs, divided into many sections, with a teaching assistant (TA), under faculty supervision, being the primary person in the lab. In our institution both undergraduate and graduate students serve as TAs. Prior to the use of **WBS**, TAs also carried out the scoring function; we found them to be conscientious and skilled. Nonetheless, there was some variability in scoring standards which were difficult to address with written and verbal instructions. In addition, especially toward the end of a term, the time requirements for scoring sometimes became a burden on the TAs. It is also true that some TAs felt that the scoring aspect made them more responsible for the course and consequently to the students.

What can be scored by **WBS**? In the current version of the program, the attributes of data which may be scored include

- the number of decimal places in a response;
- the number of significant figures in a response;
- the accuracy of a response, including unknowns;
- the accuracy of a calculation based on submitted raw data;
- the accuracy of a response in comparison with a class average or median;

We would anticipate that many other items can be scored, including chemical formula (remember: lower division labs). Additional scoring capabilities will be added to future versions of the application; suggestions for what to score will speed up the inclusion of any particular feature.

Since these responses will be scored accurately, instructors may focus on experimental technique and guiding student calculations. Students understand that the scoring is conducted impartially, and that the accuracy of the result matters. This encourages students to employ careful techniques to improve submitted results.

5 Requirements

There are some requirements which must be satisfied before **WBS** will work properly. However, these requirements are minimal and are probably already in place at your institution.

1. **Web Server supporting PHP and MySQL.** Installation of **WBS** requires the cooperation of the people who operate the web server in your institution, be it at the departmental or some higher level. The cooperation is needed in several ways. For

convenience, we will refer to these people using the currently popular acronym IT. IT must offer both PHP and MySQL. Both are already installed at many institutions, so there may be no additional technical costs. Both programs are (as of this writing) available at no cost to academic institutions, so there should be no actual financial burden either. Students must have access to a computer offering a browser, preferably during the laboratory meeting at which measurements are taken, although they could report the data after lab.

2. **Web Forms.** **WBS** involves the use of web *forms*; the server must be configured to handle these in the proper fashion. Forms are commonly processed by web servers, so server configurations normally permit these.
3. **MySQL access.** Access to a MySQL database secured with a password must be arranged.
4. **Student logins.** since students submit material which is **scored**, the application expects students to **login** to their account, in a PHP file. Again, this is common practice at most institutions, but details differ from place to place. Consultation with your IT department is therefore required. This is the **ONLY** technically demanding part of the installation.
5. **File system structure and access.** Components of the application (*i.e.*, files) are to be installed in several directories which must be accessible by your web server but **NOT** by students. These files are just text files, not executables themselves, which should reassure your IT consultants.

An example is actually simpler to understand than the wording above. At one institution all the files in the tar file are installed in a single location, just as they come from the tar file. The main directory (**wbs**) is under the hierarchy of files for the department in general. A link to the initial student file (*location/wbs/studentfiles/wbsindex.html*) is placed on the main web page for the lab course. Access to all other files used by students is then automatic.

You may wish to bookmark the initial scoring file (*location/wbs/adming/ireport.php*) for the individual(s) responsible for record keeping.

And, of course, sufficient disk space. Disk requirements are not large.

6 Installation

Installation Overview

1. Download the compressed file of your choice, either **wbs.tar.gz** or **wbs.zip**. The content of each is identical, and includes text files and this manual in pdf format.
2. expand the distribution file **wbs.tar.gz** or **wbs.zip**, whichever you are using. The resulting files must be located so that the web server can see them but students cannot.

See item# 5 above.

3. make the changes indicated in the files `dbgnames.php`, `dbexpts.php`, `dbdataitems.php`. More detailed descriptions of these changes are given below.
4. set up one file with the name `specificXX.php` for each experiment for which web based scoring is to be implemented. In this name the letter 'XX' are to be replaced by the code established in `dbexpts.php`. Each of these files is independent of the others, so they may be implemented individually rather than all at once. No such file need be generated for experiments in which **WBS** is not used.

In the box

The box — the compressed file referred to in the first step above — contains files in a number of subdirectories. Three subdirectories referred to frequently in this manual are named `studentfiles`, `adming`, and `dblocal`, but two of these names may be changed at your convenience. The web server used at your institution needs to be able to read the files in this directory, that is, all the subdirectories. (Well, actually, the browser never needs to use the subdirectory with the manual, but it won't hurt if the server could see files there.) You as the **WBS** administrator need read and write access as well. Students do not need access to any of the subdirectories except through the browser, and should **not** have access to any of these files. Another subdirectory, `cssfiles`, contains style files, which affect the appearance only.

To repeat, students must NOT be able even to read files in `adming` or in `dblocal`, and do not need read access to any wbs files or directories; only the web server (and the instructor) needs read access. In the normal course of operation, students will be able to see the directory and file names, so access to the directories other than via the web server must not be possible. Furthermore, the password to the database containing scoring information is contained here, so scoring security would be compromised if a student were able to read the files. We recommend that access to the files be limited to those with a need to administer the scoring.

Expansion of the compressed file will generate the subdirectories. You may then rename `adming`, & `studentfiles` as you wish, but the current version of **WBS** does not allow for alternate names for other subdirectories. This allows the opportunity to change the name of either or both of these two subdirectories.

`cssfiles` contains several files required for the appearance of the html files.

`dbfiles`, `dblocal` These subdirectories contain files which are accessed only indirectly, and not called explicitly; see **Figure 1**. Each must be readable by your web server, but, like `adming` itself, must NOT be accessible to students in any manner.

`adming` contains some of the files used for scoring of student submissions.

Student laboratory data submissions are collected by the sequence of files shown in **Figure 1** below. That is, `wbsindex.html` leads to a sequence of php files; students need only know

the name `wbsindex.html`, as the other names are hard coded into each file. Thus, if you wish for some reason change the name of any of these files, you would need to modify the files. This directory also contains several auxiliary files used by the php files:

- `instuct.popup.html`
- `instructgather.popup.html`

7 Configuration: Data Submission Files

Introduction

You need to configure **WBS** for the experiments which make up your course; we offer detailed instructions to guide you through the configuration process. This process is described here and in section 8.

Configuration is by far the most tedious part of the installation exercise. In our experience it takes about twenty (20) minutes per experiment; spread out over a semester this is not too onerous, and it needs to be done only before first use of a given experiment. This phase is where some pedagogical decisions are made, so perhaps it is appropriate that this is the time-consuming part.

While this step can be time consuming, it does not require any programming skills to complete. It does call upon your judgement of what data your students should be submitting in order to get a score. In our own experience, all of these decisions have **already** been made and indeed codified in order to provide an **answer key** or a **grading guide** for the graders. It then only remains to transfer this into the proper format. In carrying out this task, reference to the screens generated by the sample cases provided will make the task much easier.

In the next section we describe the process of editing a file, and saving it while retaining the proper formatting (which is no formatting). In the third section we offer detailed, specific descriptions of the changes to be made. A brief final section is at this point of development basically a placeholder for what the user needs to do to change the appearance of the various pages. We have tried to make the manual completely self-explanatory, but perhaps we have on occasion missed the mark. Please do not hesitate to contact us if you find the manual confusing or (horrors) wrong.

Editing a database file

In the next section, you are instructed to supply certain values specific to your institution. In this section, we attempt to explain *editing* a file, which is the way in which you enter these values.

The word 'database' appears in the title to the section because it describes the manner in which **WBS** handles this information. The kind of data the user needs to provide ranges from essentially trivial (the name of the experiment) to those involving pedagogical decisions (*e.g.*, which data students are to submit). In our applications, we request from students only data which is already in the laboratory manual or worksheet, and we recommend this practice. This approach also means that requests for items to be submitted have already been expressed in the laboratory manual itself.

In order to supply this information, it is necessary to *edit* a file. This means to change the contents of a file and save the change(s). A **requirement** is that the file be saved as a plain text file, which in turns means that it be saved without any of the extra formatting such as that which most word processing programs (*e.g.*, Word) insert into a file. There are many programs which can be used for editing (*e.g.*, Notepad, Wordpad, and many others) and indeed most word processors, including Word, can save files as simple text files, although not normally by default. Please note that a rich text formatted file (.RTF, .rtf) is **NOT** a plain text file.

The files for which you provide values need to have a .php suffix. If it is necessary or convenient to save it with some other suffix (*e.g.*, .txt), be sure to change it back to the .php suffix.

The specific changes which you make to configure a file are documented in the next section. We have also tried to provide the appropriate instructions in the files themselves. This is done by adding comments (characters that are NOT computer commands or data, but are instead intended to be human readable) into the file. In HTML, a comment begins with the characters <!-- and ends with the characters -->. In PHP, there are two ways to place a comment. One is to put in a double slash (`// this is a comment`). The double slash itself and everything which follows to the end of the line is ignored by the PHP processor, and so is intended for the humans looking at the file. The second kind of comment starts with the two characters `/*` and ends with a `*/`. The difference is that this second form permits a comment to extend over multiple lines.

```
/* this is  
also a comment */.
```

Since there are lots of comments in the files, this should make the task of configuring the files properly easier.

Configure three files

Three files require attention; they are: `dbgnames.php`, `dbexpts.php`, and `dbdataitems.php`. All are located in the `dblocal` subdirectory. The first, `dbgnames.php`, provides general information about the course. This will rarely need updating after its initial configuration. The second, `dbexpts.php`, provides general information about each experiment for which **WBS** is to be used. This will need updating as experiments change. The third file, `dbdataitems.php`, specifies to the students the information which each is to report to **WBS**. The editing is explained in detail next.

dbgnames.php

This file contains details for your institution, and undoubtedly some editing will be required. However, the first few lines of the file should not be changed, and only those edits described here should be made.

The first change to be made occurs in the segment of the code shown here:

A few lines from dbgnames.php

```
/*
 * file: dbgnames.php
 * USER EDITS start here
 *
 * The following 7 items need to be set to values appropriate for your
 * institution/course
 * replace the characters Chem 100 with the course ID; this appears in
 * the title printed on each page. See figure 1.
 * You may also note the use of double quotes rather than single; the
 * difference is not important.
 * If you don't change it, the course number will appear as Chem 100
 */
$Crseenum='Chem 100';
```

The code segment above contains mainly comment lines, which are ignored by the computer. The final line shown stores the course number. The user is to replace the default value listed, **Chem 100**, with the appropriate course number. Please notice that all variable names in PHP begin with a dollar sign character, and each statement terminates with a semicolon. In addition, please note the presence of paired single quotes. After user edits, the single quotes and the terminal semicolon **must** remain. If the course is in the physics department with number 110L, then the line might read

```
$Crseenum='Physics 110L';
```

after editing. There are no special restrictions; more than 8 characters may be employed, for example. While many characters might detract from the appearance of a page, it will not impact the operation of the code.

The next few lines of the code are

```
/*
 * Item 2. Specify the number of sections in the term.
 * Replace 20 with the number at your institution
 */
$numsec=20;
```

Again, comment lines are followed by the actual item to be edited. When submitting experimental data, students select their own section number from a list with maximum set by this value, and the administrative user may request scores for all sections at one time. Notice

that this value is NOT enclosed in single quotes, as the value is a number rather than a string of characters. But the statement still requires a terminal semicolon.

Two more items follow in the same format. The user specifies the number of experiments in line 30, and the number of days delay before a student may view his/her score in line 35. This value should be a small integer, and is intended to protect the integrity of unknowns when multiple sections are also spread across multiple days. It should be set low enough that scores are available to students before their next laboratory. It may be set to zero, but then students may see scores immediately. Such an approach has advantages, but in **WBS**, students may submit data as often as they wish. This avoids issues concerning typing errors, as students are always presented the data exactly as submitted before the data is saved for scoring.

Three additional items remain to be specified, all associated with the MySQL database server which stores student submitted values. The required information, in the order found in the file, is the name of the server, the user name, and user password, stored in the variables `$sqlserver`, `$sqlid`, `$sqlpwd`, as shown in the file.

```
/*
 * Items 5-7. Parameters for the MySQL database.
 * replace localhost with name of the server
 * The 'localhost' default must be changed
 */
$sqlserver='localhost';
// specify the sql username and password
$sqlid='root'; // replace root with sql username
$sqlpwd='root'; // replace root with sql password
```

Note that all three of these are character strings, enclosed in single quotes. The server name is normally in the standard URL form, something like `http://sqlserver.uruniv.edu`. This name may be as long as is needed. The user name must be provided to you by the people in charge of the server at your institution; we encourage you to use some name other than root. Your IT folks may appreciate knowing that the parameters appear in the line

```
$link = mysql_connect($sqlserver, $sqlid, $sqlpwd);
if (!$link) {
    die('Could not connect to MySQLserver: ' . mysql_error() . $ormsg);
```

and nowhere else.

This information is secure in the file, so long as students do not have access to the `dblocal` directory. The information is not contained in any form in the web page which uses `dbgnames.php`, so long as the php file is not changed except for the edits required here. However, it is necessary that the file be protected from unauthorized access; even the file names should not be available outside of those installing and editing the files.

A summary of the required course information is present in Table 1.

Table 1: Required installation specific parameters

line #	Code Variable	Purpose
20	<code>\$Crseenum</code>	specify course ID
25	<code>\$tnumsec</code>	number of sections in the course
30	<code>\$tnumexpt</code>	number of experiments in the course
35	<code>\$daysdelay</code>	minimum days a student may be view score
41	<code>\$sqlserver</code>	name of mysql server
43	<code>\$sqlid</code>	userid for the sql server
44	<code>\$sqlpwd</code>	password for the sql server

Nine (9) additional items which follow may be optionally changed. The final seven of these affect the appearance of the web pages. These parameters, `$Ticlr`, `$Ditembg`, *etc.*, specify certain colors used in the application. I recommend leaving them for now, so your screens will look just like the figures here, but then feel free to change them to your own taste. Documentation is not currently available. Other lines of code should not be changed.

`dbexpts.php`

In this file the user must supply identifying information for each experiment. Each experiment is independent, and missing entries for one experiment have no effect on another, so experiments may be entered as needed. In the sample file provided the values described below are arranged by experiment. All items to be changed follow the line

`//User changes below`

which, being in a php portion of the file, is a comment. The file provided includes a complete set of information as used at one university at one time. There is no expectation that the values will be useful to you; they are included merely for illustrative purposes. It would be best to save the supplied file with a different name for reference purposes, and replace the values here with the values for your course.

A few lines from the file are

```
//  _____ EXP #3 Conservation of Mass _____
$erefno=3;                      // integer by which expt referenced
$abbrev='MA';                   // symbol by which expt referenced
$name[$abbrev]='Mass Relations '; // TITLE
$eabbrev[$erefno]=$abbrev;
$nod[$abbrev]=6;
```

They items needing your attention (for EACH experiment) are

\$erefno = n, where n is simply the sequence number of each experiment, the order in which the experiment is performed, beginning with 1 and ending with the value of `$tnumexpt` as provided in `dbgnames.php`. This is not provided automatically because it could change from term to term. The user needs to specify only the integer value following the equal sign.

\$abbrev= 'XX' , where the user must replace the characters XX with a unique, two character abbreviation which identifies a particular experiment. This information is a *string*, not a number as the preceding case, and in php must be enclosed in quote marks, as in the examples.

This seems to duplicate the **\$erefno** information, but the sequential order of experiments might vary from term to term, while our intent is that the abbreviation be permanent. The default abbreviations in the file are unique and may be left as is, but the intent is for the abbreviation to serve the programmer as a short mnemonic for the specified experiment.

This value must be specified before the ones which follow.

\$ename[\$abbrev]='title' is the title of the experiment, preferably the same as the title in the laboratory manual. For our purposes, single quotes are generally preferred, but double quotes will work, too. Again, the user replaces only the five characters inside the quotes with the title of choice, which may be more than 5 characters. It is displayed to the student on each web page during the submission process.

\$nod[\$abbrev] = n, where n is to be replaced with the number of data items requested of students for that experiment. The value must be a non-negative integer. Each sequence number should be specified; set **\$nod=0** to skip a given experiment. In this case no further values for the experiment need be specified, but students will benefit if the title is still specified. In this example, the value assigned is 5, which means that students will encounter requests for 5 pieces of data. What the requests are the user specifies in `dbdataitems.php`, described below.

\$eabbrev[\$erefno]=\$abbrev should be included for each experiment to be used in **WBS**.

No other items in the file need to be changed. Indeed they should not be changed.

Finally, we note that each experiment stands alone, so that if the file has information only for experiments 1, 3, and 9, the application works fine for experiments 1, 3, and 9.

Table 2: Required Experiment Parameters

Code variable	Purpose
\$abbrev	unique two character identifier
\$erefno	sequential order of the experiment
\$ename	title of the experiment
\$nod	number of data items to be submitted by students

`dbdataitems.php`

This file contains the requests for student data for all of the experiments, **\$nod** items for each experiment, as specified by the user. Each request is an entry in a single, large PHP array. There is no expectation that any of the specifications in the sample file will be of any use to the user; they are included purely as examples. The format is straightforward; we

include a few lines from the sample file for reference:

```
"OX1"=>"best value , molarity KMnO<sub>4</sub>",  
"OX2"=>"best value , molarity H<sub>2</sub>O<sub>2</sub>",  
"OX3"=>"% H<sub>2</sub>O<sub>2</sub>",
```

Each line contains two items enclosed in double quotes, separated by two symbols: =>. The first item begins with the mnemonic (**\$abbrev**) assigned to the experiment, followed by the integer sequence number of the request. In the example, the mnemonic is **OX**, and nod was specified as 3, so the entries are **OX1**, **OX2**, and **OX3**. There must be no blanks in the characters to the left of the '=>'. These three or four characters are enclosed in the first set of double quotes. This is followed by the two characters =>. The second item is the text of the request enclosed in double quotes. This text will be presented to the students, as shown in **Figure A**.

Please note that the request may contain HTML code. In this example units are indicated by changing the color to green; also, a simple chemical formulas are rendered properly with subscripts. Superscripts may be presented similarly, with ^{and} enclosing the superscript.

This is some work, but the real effort is in determining the information which should be requested of the students.

```
Summary for dbdataitems.php: specify values for the array.
```

Appearance

The appearance (fonts, colors) of **WBS** is determined by Cascading Style Sheet (css) files located in the **css** directory. At the moment the user has a choice of only one style; perhaps this can be increased in future if there is interest. Of course, the user may alter the style sheets as desired.

8 Configuration: Scoring Student Data

A second set of programs scores the data submitted by students. The sequence of scoring files is shown in **Figure 1**.

As shown in the figure, scoring begins by browsing to the file **ireport.php** in the appropriate directory. The initial screen offers three choices to the user. The one of interest here is the default choice, **Scoring**. Thus, all the user need do to score is click the submit button. The remaining options are explained below in subsection "Other Options" on page 22.

All the real work is done by the code contained in the file named **scoreengine.php**; it should not be modified unless the user is thoroughly familiar with the coding. Of course,

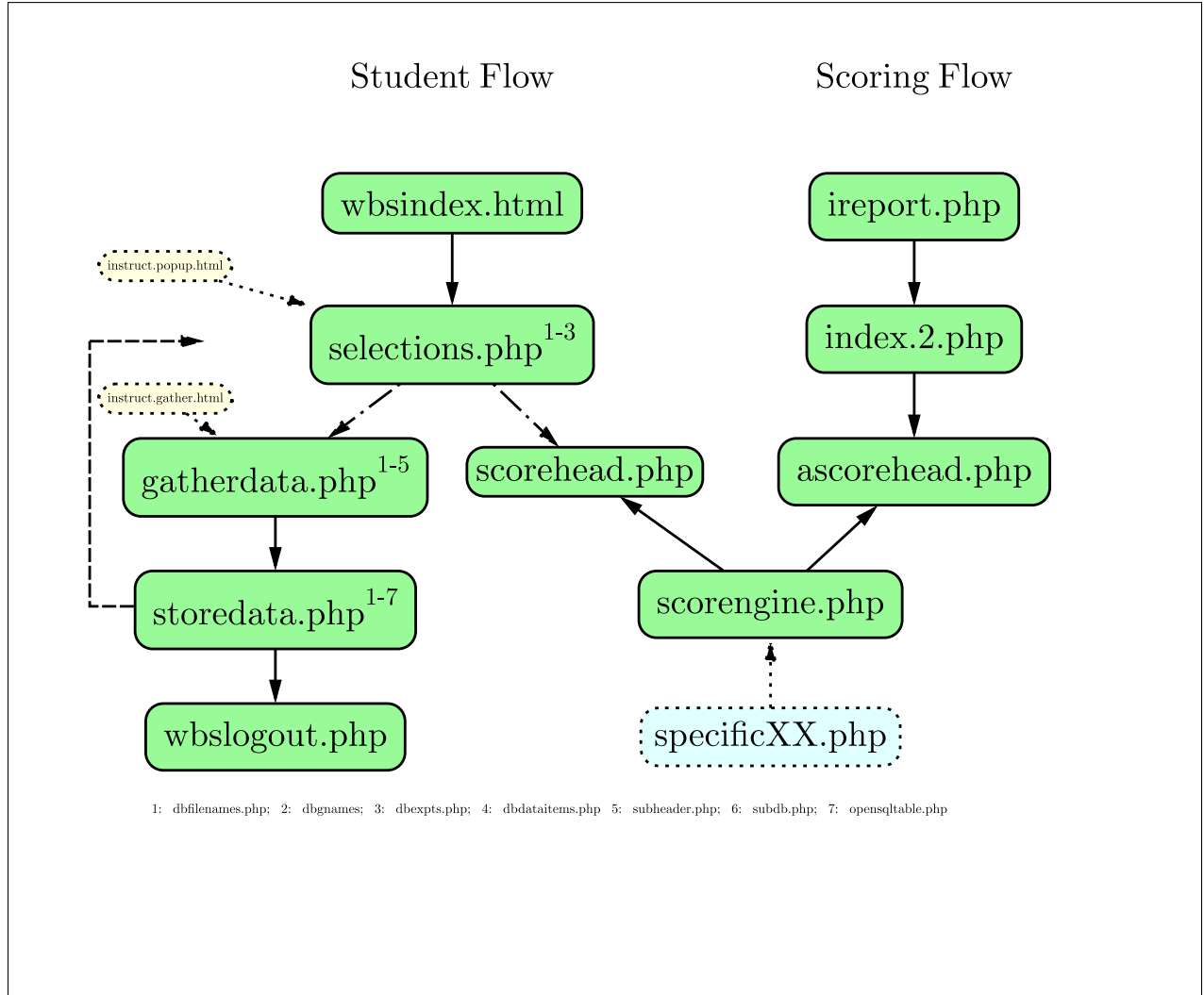


Figure 1: Flowchart of program operation. Data submission follows the leftmost sequence. Instructor scoring follows the rightmost. Score reporting to a student starts in upper left and flows across at the dot-dash line. Please note that the files `scorehead.php` and `ascorehead.php` are stubs which define a single parameter and then use `scoreengine.php`, which does all the work of scoring.

each experiment has different data and is graded uniquely; these conditions are brought into the code via the configuration specified separately for each experiment in the a named `specificXX.php`, where XX stands for the mnemonic specified in the previous section. Thus, if nine experiments are to be scored by **WBS**, there must be nine files with the proper, unique choices for XX. This section is concerned with the generation of these files. Several examples of actual configuration files are included with the distribution, and a further file, actually named `specificXX.php`, contains an example of each type of scoring possible in the current version.

Thus, setting up the configuration is straight-forward. As in the case of the data submission configuration, one file is needed for each experiment, so it is admittedly somewhat tedious. And also as in the experiment configuration files, the non-existence of a `specificXX.php` file for one experiment will not impede the operation of scoring for other experiments.

In our own implementation of **WBS**, we made essentially *none* of these decisions; rather, we simply accepted the decisions which had been made for scoring by hand, based on the keys provided for TA graders. We recommend this approach as a good way to start.

In this context, it is unnecessary to request that all items recorded in the laboratory notebook be submitted via the **WBS** system; only those data items which must be graded themselves are requested. In addition, we also require those items which are needed for the scoring of other items; that is to say, we can and do use the computer to check calculations. It seems to us that this is a true advantage of the computerized scoring compared to manual scoring, unless your TA's have the time and take the care to check calculations themselves (good for you and them).

There is however no inherent limit in the number of items requested or graded in the **WBS** programs. As an example, in one of the last experiments of the term, students are to determine the concentration of an unknown; we request only two items from the student, (1) the unknown number and (2) the concentration. In an earlier (and easier) experiment, we ask for eight (8) items and score them all.

What can be scored?

We think virtually anything that can be put on the scoring key.

Numeric input

For any piece of numerical data, in the current version the following can be marked:

1. number of decimal places
2. number of significant digits. At least, based on our definition of significant digits. It might be useful to give our rules:
 - (a) any digit in a number is a significant digit, except

- (b) a leading zero

This is the simplest definition of which we are aware, and avoids all arcana.

3. value of a result, compared to

- (a) a prespecified value;
- (b) or a value computed from other submitted data;
- (c) or a value based on a class “average”

In each case, the result is not a simple right or wrong, but rather a gradation based upon the difference between the student’s reported value and the “correct” value.

The author has been informed by faculty much more skilled at the evaluation of analytical results that scoring of unknowns based on concentration or the equivalent is a very difficult undertaking, because there are a number of variables difficult to control in the student laboratory setting. These extend down to the level of insufficient mixing of unknowns by laboratory faculty or staff. **WBS** gives the instructor the option to look at all student results, and to compare the results for an individual with the whole set of student results. We have found this to be effective.

Non-numeric input

Actually, we haven’t tried this yet. But there is certainly no reason that a simple chemical formula, for example, could not be scored.

How to instruct **WBS** to score

The files `specificXX.php`, located in the administrative subdirectory, contain the details of what is to be scored, and how it is to be scored. First, and most easily, the characters ‘XX’ in the file name are to be replaced with the mnemonic used for a given experiment; these are capital letters if you define the mnemonic with capitals. Thus, eventually you will need to generate one file for each experiment. There is actually a file named `specificXX.php` in the distribution; this file contains a template of each type of scoring in the current version of **WBS**.

The `specificXX.php` files which you create should be located in the `dblocal` directory.

Again, I recommend opening one of the example files while reading these instructions. In fact, these remarks are tailored to one of the examples provided, `specific02.php`.

Each item submitted by students may be marked on the basis of any or all of the “attributes” listed below. By default, each attribute is turned off, and so ignored (not marked for that attribute) unless specifically turned on by the user. For example, line # 8 of `specific02.php` turns on marking of decimal places for the second item submitted by students.

```

$k = 2;           // specify scoring attributes for the second item
$ck_dec[$k] = True; // check # of decimal places of the second item

```

k having being assigned the value 2 in the preceeding line. Each available attribute may be turned on in a similar fashion.

The attributes available in the current version are:

number of decimal places In our manual, the proper number of decimal places required is often stated explicitly. Or it may be implicitly defined by an instrument. For example, since we use typical top-loading balances with covers, three decimal places are appropriate for masses in grams. The number of decimal places for a proper response must be specified; the complete configuration is

Table 3: Configuration parameters for decimal place attribute

variable	set to	comment
k	integer: item #	identifies which submitted value in question
$ck_dec[k]$	True	set True to score decimal places for item # k
$ndcp[k]$	varies	integer; correct number of decimal places
$wtckd[k]$	varies	deduction if incorrect # of decimals reported
$lmsgckd[k]$	a brief message printed when student checks score; <i>e.g.</i> , 'in the volume'	
preface	incorrect # of decimal places	

Here, as in the following tables, “preface” refers to a short message preceeding $lmsgckd$ or the corresponding variable for the other attrbutes.

number of significant digits The number of significant digits in a response may be determined and compared to the number reported by the student. We use the definition that any and all digits (including zeroes) reported are significant, except leading zeroes, in order to handle values less than one in specified units. The complete configuration is

Table 4: Configuration parameters for significant digit attribute

variable	set to	comment
k	integer: item #	identifies which submitted value in question
$ck_sf[k]$	True	set True to mark significant digits for item # k
$nsf[k]$	varies	integer; correct number of significant digits
$wtnsf[k]$	varies	deduction if incorrect # reported
$lmsgnsf[k]$	a brief message printed when student checks score; <i>e.g.</i> , 'for molar mass'	
preface	Incorrect # of significant digits	

accuracy Any item submitted may be checked for accuracy by comparison with a pre-defined value. The complete set of configuration parameters is given in Table 5. Implementation of scoring for accuracy was originally based on a deduction based

Table 5: Configuration parameters for accuracy attribute

variable	set to	comment
\$k	integer: item #	identifies which submitted value in question
\$ck_acc[\$k]	True	set True to score accuracy of item #k
\$corval[\$k]	varies	correct value
\$scalef[\$k]	1	used to control deduction
\$mxdeduct_acc[\$k]	varies	maximum deduction
\$lmsacc[\$k]	a brief message printed when student checks score; default value: in accuracy	

on predetermined ranges. It was found easier to present this in the form

$$deduction = \left(\frac{|xarg - corval|}{scalef} \right)^{1.25}$$

rounded to an integer. The power 1.25 in the formula means that the deduction grows more rapidly than linearly as the size of the error increases. The factor **scalef** determines the size of the deduction. The factor **mxdeduct_acc** determines the maximum possible deduction. A figure is provided to allow the user to estimate reasonable values for this quantity to put **WBS** in close agreement with current scoring scheme. Of course, if a more sophisticated analytic method is used, that could be coded into the application, and **WBS** would be happy to assist in the implementation of any such.

It is also possible to score based on accuracy if the correct value is not known, but if the correct value can be based on something like the median value, to help account for year to year variations in solutions, environmental samples, and the like. This requires only a little more work for the user.

value of an unknown is handled as a separate attribute, but has many of the same features as scoring for accuracy. The main difference is that a table relating unknown # to result must be constructed; see the example. It occurs inside a function of the name specified in the example.

Table 6: Configuration parameters for unknown attribute

variable	set to	comment
\$k	integer: item #	identifies which submitted value in question
\$ck_unk[\$k]	True	set True to score unknown value of item #k
\$unkid	varies	item which request unknown identification value
\$mxdeduct_unk[\$k]	varies	maximum deduction
\$lmsgunk[\$k]	a brief message printed when student checks score	

computation Often, students are requested to compute a value (the molar volume of oxygen at STP, for example). The result may well be implicit in data submitted. The application may be instructed to compute the value correctly, and take a deduction

should the value provided by the student be incorrect. The properly computed value may then be used to check for accuracy.

Table 7: Configuration parameters for computation attribute

variable	set to	comment
\$k	integer: item #	identifies which submitted value in question
\$ck_comp[\$k]	True	set True to score computed value of item #k
\$compenalty	varies	deduction
\$lmscomp[\$k]	a brief message printed when student checks score	

value compared to class average involves (in this version) more configuration parameters than any other attribute.

What can't be scored?

There is probably a long list. For example, we require that students use ink to complete the “laboratory notebook” and require that incorrect entries be legibly struck out with a single line and the correct entry made nearby. At one time we scored student papers about these matters, which do not translate nicely to **WBS**. We no longer score on this basis, but rather ask TA's to check on this and have students make corrections as the TA moves around the laboratory. This works at least as well, if not better, than scoring these attributes.

Other Options

The initial instructor screen (`ireport.php`) presents three options to the user, selected by clicking the appropriate radio button. The first and default option (**Scoring**) is to score lab reports for one or all sections, as explained above. The other options allow the user to verify his configuration of **WBS** as described in Section 7 and Section 8.

The second option, (**Verification**), prints out the configuration items not specific to a given experiment, such as the course name and number. The third option, (**Test**), prints out the configuration for a given experiment, including requests for data input which **WBS** will present to the students.

9 Technical Details

Introduction

The section of the manual is NOT needed either for installation or for operation. Rather, it is for programmers who wish to see how the coding works. It may also be used by those

who wish to extend capability, which can be done easily by adding the appropriate PHP functions and/or calls.

Design Principles

Every experiment is different, seeking to expound different experimental techniques to students, and to illustrate different scientific principles in support of lectures. After several false starts, we concluded that attempting to construct a suite of programs was a maintenance nightmare, and not feasible, at least not here. Instead we found it possible to have one “program”, controlled by external databases which provide the specialization needed. After all, checking the number of decimal places or the number of significant figures does not depend on the chemistry of the experiment.

As mentioned above, by the word “program”, we generally mean a suite of one or more PHP files which carry out a specific task. The files are typically web forms which call other files in a specific sequence. For example, the data collection “program” consists of one plain html and four PHP files. We list them in **Table 8** to be specific. This approach also ensures

Table 8: Data Collection Program Files

wbsindex.html	welcome screen; submit data or check score
selection.php	form collects student login, lab section, and experiment ID
gatherdata.php	collects experimental data
storedata.php	echos data for verification or correction by student
wbslogout.php	logs out student, stores data in MySQL file, reloads selection.php

that the student is presented with the same format every week, so that an initial lack of familiarity can be rapidly overcome.

The program requests that students supply data, always in exactly the manner in which it is entered in the standard report. Thus, the use of **WBS** requires data submission, but no additional experimentation or calculation on the part of the student. All student data is stored in the form of a string of characters, even though most or all of the data is numerical in character. This ensures that all data stored and available to the scoring program and the grader is *exactly* what the student typed.

Known Bugs

1. the method to acquire user names is flimsy, and will not work properly if the username contains a blank (*e.g.*, Van der Waals).

Notes on Functions

chkacc(xarg, corval, scalef, maxdeduct, lmsg)

Purpose

Determine a deduction based on how close xarg is to corval;

Arguments:

scalef must be positive

Definition

$$chkarg = \left(\frac{|xarg - corval|}{scalef} \right)^{1.25}$$

Output

The function returns a integer value between 0 and the value supplied for maxdeduct, inclusively. The value of the function reaches 10 when $\frac{|xarg - corval|}{scalef} \approx 6.1$, as may be seen from the figure. This should be used to guide the user in determining the proper value of *scalef*.

Please see the related function *chkacc2* which can provide a steeper (or less steep) increase with the size of the error.

A quick primer on the use of php files

Here we give a brief explanation of the design issues involved with use of php files. A (dot)PHP files is meant to be handled by a web server, much in the same way that a (dot)html file is handled. There is an important difference; the PHP file is first treated by a preprocessor. A PHP file may contain html commands; these are simply emitted to the web browser which called up the PHP file. So if a (dot)PHP file contains nothing other than html statements, the result is identical to that elicited by a (dot)html file, except for the wasted processor time used by the server.

The point of a PHP file is to contain PHP specific statements, which are treated according to the rules of the PHP language. The PHP statements are themselves NOT SENT to the web browser which called up the PHP file. Instead, only those items explicitly sent by specific PHP statements are sent. Thus, a PHP file may contain a password, but that password is not visible to the user of the web browser (unless, of course, the person writing the PHP file foolishly sends it to the browser).

In the files provided here, PHP files not only contain passwords, but they carry out calculations and comparisons, based on the data supplied by the students and by the instructor.

Examples often make things clearer. The first file listed below in an appendix, `wbsindex.html`, contains no PHP statements, as the suffix indicates. This short file does show a simple form in use (if you can squint hard enough to read it).

```
<!DOCTYPE html>
<head>
  <meta name="generator" content=
    "HTML Tidy for Mac OS X (vers 31 October 2006 - Apple Inc. build 15.3.6), see www.w3.org" />
  <title>WBS Welcome page</title>
  <meta http-equiv="Content-Type" content="text/html; charset=us-ascii" />
  <link rel="stylesheet" type="text/css" href="../cssfiles/form.css" />
</head>

<body>
  <h2>Welcome</h2>

  <p>to web based scoring of laboratory data. You may either</p>

  <ul>
    <li>submit data you recorded during the laboratory experiment,
      or</li>

    <li>look up scoring of your earlier submission.</li>
  </ul>

  <p>Simply click on the appropriate radio button below, then click
  on the button labeled "Continue".</p>

  <form name="form0" method="post" action="selections.php" id="forma">
    <p><input type="radio" name="sorg" value="datain" checked=
      "checked" /> Submit lab data<br />
    <input type="radio" name="sorg" value="student" /> Check your
    score<br /></p>

    <p><input type="submit" name="Submit" value="Continue"
    class="button" />
    [Press (click) the button to the left to continue]</p>
  </form>

  <!-- <p><font size=3>Copyright 2010 WBS Enterprises</font> -->
</body>
</html>
```

The second file, `selections.php`, does contain PHP statements, flagged to the server by being enclosed by “`<?php`” and “`?>`” characters.

```
<!DOCTYPE html>
<head>
  <meta name="generator" content=
    "HTML Tidy for Mac OS X (vers 31 October 2006 - Apple Inc. build 15.3.6), see www.w3.org" />
  <meta http-equiv="Content-Type" content="text/html; charset=us-ascii" />
  <title>WBS Login, Select Lab</title>
  <link rel="stylesheet" type="text/css" href="../cssfiles/form.css" />
  <script type="text/javascript" src="popup.js"></script>
</head>

<body>
  <!--
    second file in the data collection process
    gathers course, section, and experiment #
    named output: firstname,lastname,sec,expt, sorg
  -->

  <h2><?php
  $m=' does not exist. Either reinstall WBS or contact the wbs webmaster.</h2>';
  define ("Directcall", true);

  (@include '../specials/checkorder.php') OR die('<h2>The file ckorder' . $m);
  $t= ckorder('wbsindex') OR DIE($ord.msg); //kill if improper caller

  (@include 'dbgnm.php') OR die ('the file g ' . $m); // general defs
  (@include 'dbgexpts.php') OR die ('the file ex ' . $m); // expt names

  //Grab the posted data
  $which = $_POST['sorg']; // possible values of $which: datain, student
  echo "<!-- Input value:  $which -->\n";

  if ($which=='datain'){
    $greet='Submission';
    $filename='gatherdata.php';
```

```

} elseif ($which=='student'){
    $greet='Scoring';
    $filename="scorehead.php";
} else {
    die('Oops. You seem to have stumbled on this file by mistake.<br>
        Perhaps you wanted <a href="wbsindex.html">this</a>');
}

echo "$Crseenum Labs<br>
    Web $greet for Experiments</h2>\n";

$uid=posix_getuid(); $uinfo=posix_getpwuid($uid);// print_r($uinfo)
$fullname=$uinfo[gecos];
echo "<!-- user name is $fullname -->\n";
$eullname=explode(" ", $fullname);// print_r($eullname)
$name=$eullname[0];
$lname=$eullname[1];
$tmp=$SERVER['DOCUMENT_ROOT']; echo "<!-- doc root $tmp -->\n";
$tmp=$SERVER['PATH_INFO']; echo "<!-- path info $tmp -->\n";
$tmp=$SERVER['HTTP_HOST']; echo "<!-- http host $tmp -->\n";

echo '<form name="form1" method="post" action="'. $filename .' " id="form1">'. "\n";

// submit first & last names to the form, no blanks in last name
/*
* replace with rawurlencode
echo "\n<input type=hidden name=lastname value=\"";
$nmelength=strlen($lname); //echo "<p> # chars in last name is: $nmelength";
for ($i=0; $i<$nmelength; $i++) {
    $tc=$lname[$i];
    if ($lname[$i]===" ") $tc="_";
    echo $tc;
}
echo "\">\n";
echo "<input type=hidden name=firsnme value=\"".$fname.">\n";
*/
echo "\n<input type=hidden name=lastname value=\"".rawurlencode($lname)."\">\n";
echo "<input type=hidden name=firstnme value=\"".rawurlencode($fname)."\">\n";
echo "<input type=hidden name=sorg value=$which>\n";
echo "<input type=hidden name=output value='html'>\n";

echo "<p>Hello, ".$fname." ".$lname.". ";
?> Please select the proper entry in each column:</h2>

<fieldset>
    <legend class="cl">Lab Section # and Experiment #</legend>

    <p>Section: <select name="sec" size="4" class="slctlist">
        <?php
            for ($i=1;$i<=$numsec; $i++) {
                echo '<option value="'. $i .' "> $i \n';
            }
        ?>
    </select> Experiment <select name="expt" size="4" class="slctlist">
        <?php
            for ($i=1;$i<=$numexpt; $i++) {
                $eid=$eabbrev[$i];
                echo '<option value="'. $eid .' "> $i $ename[$eid] \n';
            }
        ?>
    </select></p>
</fieldset>

<p>[Press (click) the button to continue] <input type="submit"
name="Continue" value="Continue" /></p>

<p>(Click <a href="instruct.popup.html" onclick=
"return popup(this, 'instructions')">here</a> if you wish to
review basic instructions.) <span class="c3">spacing</span>
<?php echo $cn; ?></p>
</body>
</html>

```

The user (a student) is asked to select both a section number and the experiment for which data is to be submitted. The file does NOT check to make sure that the student has made valid selections. All student input, as read by the computer and including the section number supplied on this page, is echoed on a confirmation page, which affords the student the opportunity to correct any miscommunication.

Appendix A Screens for Data Submission

Wbsindex.html

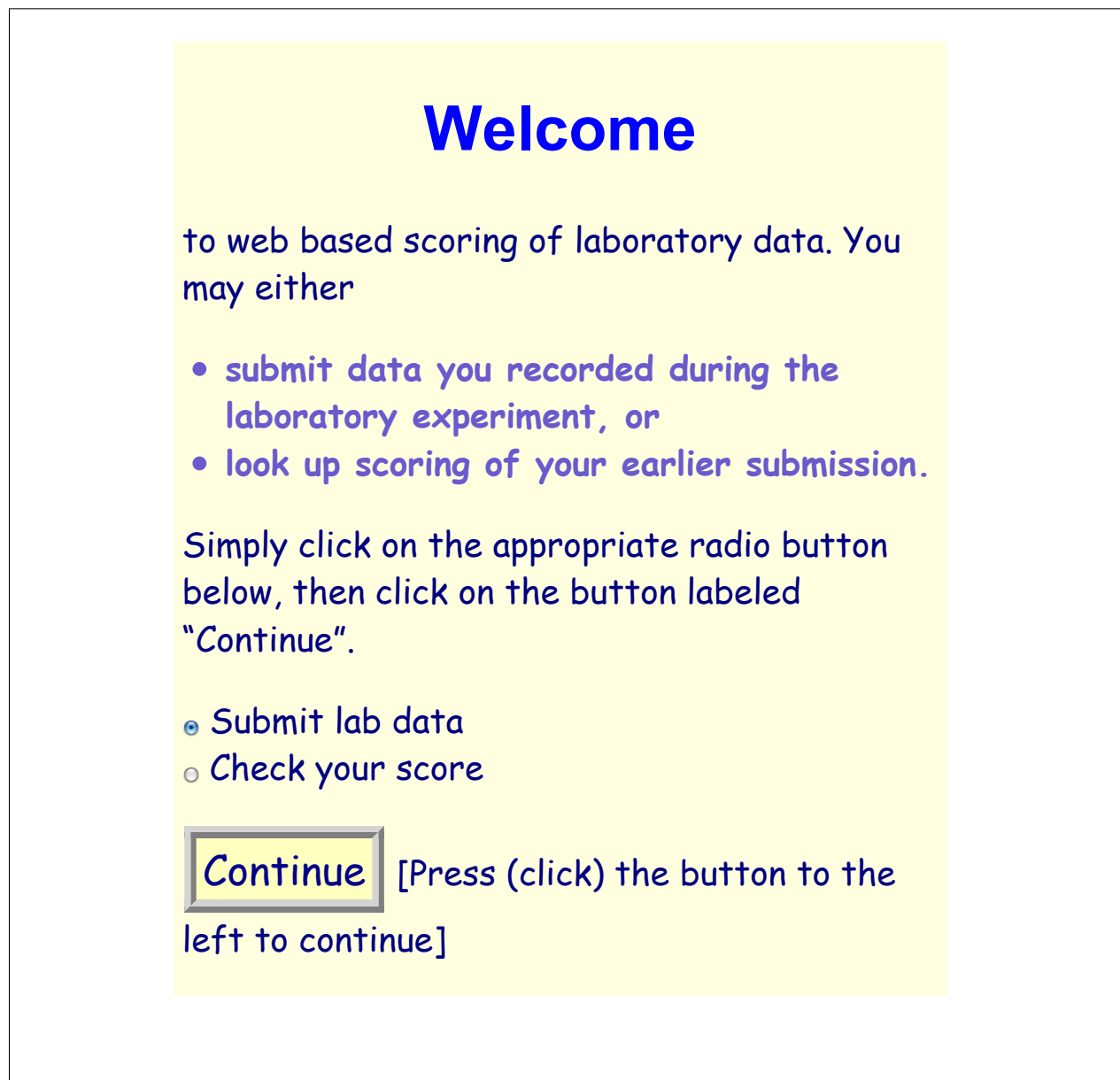


Figure 2: Initial student screen

Chem 100 Labs

Web Submission for Experiments

Hello, Stephen Knudson. Please select the proper entry in each column:

Lab Section # and Experiment #

Section:	<div>1 2 3 4</div>	Experiment	<div>1 Introduction 2 Avogardo's Number 3 Mass Relations 4 Acid/Base Titration</div>
----------	--------------------------------	------------	--

[Press (click) the button to continue]

(Click [here](#) if you wish to review basic instructions.)

Copyright 2010-12 WBS Enterprises; Version: 1.4

Figure 3: Students pick both section number and experiment.

Chem 100 Labs

Expt. # 4: Acid/Base Titration

Please enter the requested quantity in the spaces below.
Enter ONLY numbers (which may or may not be integer),
NO units.
For scientific notation, use E or e: *e.g.*, 3.14e+4, -
2.718E-3.
All data is on your report sheet(s).

Average M of NaOH solution	<input type="text"/>
Average % acetic acid in sample	<input type="text"/>
Part III,top: # millimoles H ₃ PO ₄ (nearest 0.01)	<input type="text"/>
integer k, BCG indicator	<input type="text"/>
integer k, TPL (thymolphthalein) indicator	<input type="text"/>

press the button to submit your data:

(Click [here](#) if you wish to review basic instructions.)

Copyright 2010-12 WBS Enterprises; Version: 1.4

Figure 4: Typical experimental data submission screen. Note that units are predefined.

Storedata.php

Confirmation Page

Expt. # 4: Acid/Base Titration

Name	Stephen Knudson
Course - Section	Chem 100 - 2
Experiment #	(see above)
Average M of NaOH solution	1
Average % acetic acid in sample	2
Part III,top: # millimoles H_3PO_4 (nearest 0.01)	3
integer k, BCG indicator	4
integer k, TPL (thymolphthalein) indicator	5

Wed, Oct 17 2012, 10:10:36 AM

Thanks for submitting this data.

(If you need to make a correction, just click the Back button near top of the window)

If your values, shown in red, are correct, click [LOGOUT](#) to finish, or
click [HERE](#) to start over.

Copyright 2010-12 WBS Enterprises; Version: 1.4

Figure 5: All data entered by a student is echoed for confirmation.

Appendix B Computer Requirements

Here we summarize the requirements for your institution and students.

In order to submit data, students must have access to a computer which operates a web browsers. Any of the usual ones is adequate. The computer may be supplied by either the student or the institution.

Your institution

- must operate a web server which
- supports forms and
- PHP.
- must operate a MySQL database.
- must support a student login for security.
- must provide files readable by the web server but not by students. These files should have read/write permissions for the instructor. (Note: the `php` files in these directories are NOT trivially read from the html on the student's browser, as only selected portions of the `php` files `php` are sent to the browser.)

Appendix C Version 1.6

This manual is specific to version 1.6, although changes from version 1.4 do not directly affect student files. Major changes are the addition of screens which allow the user an easier means to verify user edits, as described in Sections 7 and 8. The code remains largely html5 compliant, according to the verification tool at <http://validator.w3.org/check> and improved CSS support, verified at <http://jigsaw.w3.org/css-validator>