Web Survey Mailer System (WSMS 1.1)

Ernesto Barrios
Center for Quality and Productivity Improvement
University of Wisconsin-Madison
610 Walnut Street 575 WARF
Madison, WI 53726

WSMS1.1 was developed at the Center for Quality and Productivity Improvement, at University of Wisconsin–Madison, as research tool for the Information Technology Workforce Project (ITWF, 2002) supported by the NSF Information Technology Workforce Initiative (NSF EIA-0120092).

For more information:
Ernesto Barrios
Tel: +1-608-265-2658
Fax: +1-608-263-1425
ebarrios@engr.wisc.edu

November 2003

CQPI Technical Report No. 186

Center for Quality and Productivity Improvement
Pascale Carayon, Director George E. P. Box, Director of Research
575 WARF Building University of Wisconsin–Madison 610 Walnut Street Madison, Wisconsin 53726 USA
608/263–2520 Fax: 608/263–1425 Email: cqpi@engr.wisc.edu
http://www.engr.wisc.edu/centers/cqpi

Web Survey Mailer System (WSMS1.1)

Ernesto Barrios
Center for Quality and Productivity Improvement
University of Wisconsin–Madison

November 17, 2003

Contents

List of Figures

Abstract

Nowadays, with the extended access to computers and more particularly to the Internet, web–based questionnaires are another tool available for sampling surveys. This document describes the use of the Web–based Survey Mailer System 1.1 (WSMS1.1), a computer package that administers surveys over the Internet. Created by David J. Solomon, WSMS consists in a bundle of HTML and PHP scripts that sends out personalized emails inviting people (in a database) to fill out questionnaires while anonymously tracking the individuals, so reminder emails are sent only to persons who have not submitted the survey. For any particular application WSMS scripts have to be modified. WSMS1.1 was developed in a modular form so changes to the code are localized to only few scripts. This layout facilitates the survey administration, clearly separating the tasks of the server side of the survey from the respondents side (the questionnaire and some information web-pages). In addition, a survey consent web–page has been included for the respondents explicitly agree or refuse to participate in the survey. Written more like a manual, however, this document explains some of the basics about web–servers and support software, so researchers with little experience with computers and systems and limited budget can have web–based surveys as another tool available for their investigations. WSMS1.1 ran successfully under both Windows and Linux operation systems.

1 Introduction

For more than a century sample surveys have been used in Marketing and Social Sciences. More recently surveys are being used in Education and Industrial Engineering as a tool available to scientists in their research. Nowadays, with the increased access to computers and more particularly to Internet, web–based questionnaires clearly rise as another option for carrying out sampling surveys. Advantages and limitations of the various types of surveys, including web–based, are discussed for example, in Solomon [1] and Hoonakker et. al [2]. The Web-based Survey Mailer System (WSMS), as its name indicates, is a computer package for carrying out surveys over the web. This document explains the use of WSMS version 1.1 and discusses some details. This is a technical paper written as a manual but assumes minimum computer knowledge by the potential WSMS user, explaining basic ideas of software, computer languages and Internet. WSMS1.1 was developed mainly to make the system simple and usable by researchers with little experience with computer systems and so to provide web–surveys as another investigative tool available in their research.

The Web–based Survey Mailer System (WSMS) sends out personalized email cover letters inviting people to respond to a questionnaire. The system anonymously tracks those individuals who have returned the survey so that reminder emails are sent only to those individuals who have not answered the questionnaire. Developed by David J. Solomon (2001) ([1]), at Michigan State University, WSMS consists in a set of HTML and PHP scripts that may be run in any web server running the PHP interpreter and MySQL database manager[1]. These terms are explained in section 2.

Solomon's WSMS is open source and it is available under GNU/GPL license ([3] [7]). Among some other issues, to be an open source system means that the code is available to the public. For any particular application of WSMS, it is necessary to modify the original scripts. This document describes our experience using the system and explains the extension done into what we call WSMS1.1.

WSMS1.1 works under the same idea as WSMS. It is open source and has been organized so necessary changes to the code are localized in only a few scripts. This new layout makes the administration of WSMS easier for people less knowledgable about computers. In addition, a consent form has been included as the first webpage presented to the potential respondent, so those who do not agree to participate in the survey will not receive reminder emails. Also, since MySQL is already used to handle names and email addresses, and to flag submitted questionnaires, it is now used to store the responses too. Screen shots and detailed description of the WSMS1.1 are presented in the rest of the document.

In the next section, software and system requirements needed to run a web-based survey are discussed, followed by a brief description of WSMS. Section 4 introduces what is new in the WSMS1.1 system and discusses the administration part and the respondent side of the system in subsections 4.2 and 4.3. Some details of MySQL table contents and HTML elements are included in section 5. Finally a summary of the composition of WSMS1.1 is presented in section 6. A brief description of each of the scripts and files are included in Appendix 6.

[1] You can find all the hardware/software terminology used in this document and much more in Webopedia (`http://www.pcwebopedia.com`).

2 Requirements for a web based survey

To carry out a survey through the web, the potential respondents of the questionnaire should be able to find the survey on the web and the server should be capable of capturing their answers.

To navigate the World Wide Web, a computer connected to the Internet and a browser are required. A *browser* is the application software (Netscape, Internet Explorer, etc.) able to connect to the different sites and to interpret the information provided by those sites. The communication protocol is http (hypertext transfer protocol), a suitable language for transmitting information through the net. In order for the site's computer, called the *server*, to understand the visitor's requirements it needs a program that precisely understands http. One of such programs is Apache. A visitor, called *client*, contacts the server, Apache interprets the request and returns answers to the client.

When a respondent visits the WSMS site to access a questionnaire, the server should provide the necessary information for the respondent's browser to interpret it as a form to be filled out. These questionnaires are written in HTML (hypertext markup language), a language understood by browsers. On the server side, the computer needs to know what to do with the information provided by the client. One-way of doing this is using the PHP (hypertext preprocessor) interpreter that allows the server to perform tasks by client demand in real time. PHP, in particular, also facilitates the communication with MySQL, the relational database manager used by WSMS. See the references for useful software at the end of this document([4] [5] [6]).

We have mentioned various programs and applications for different purposes. Of course, these are not the only possible choices. For example, in the Open Source[2] side there is Xitami, another web server application. Alternatives to PHP could be the use of Phyton or CGI-Perl scripts, and there is the object-oriented PostgreSQL database manager that could be used instead of MySQL. On the commercial side, Microsoft for example has applications for running web servers and managing databases. We decided in favor of Apache–PHP–MySQL for various reasons[3]: a) WSMS scripts are in PHP already, which, as we mentioned before, communicates very well with MySQL; b) the high quality of these applications is well known; c) there are versions of the software for the two most popular operation systems, MS Windows and Linux; d) the software is open source; e) since the software is free, the web based survey *tool* may be accessible to everyone under budget contraints. Early versions of our extension WSMS1.1 were tested in a PC under Windows 2000 SP3. The final version is currently running in the same PC but under Red Hat Linux 7.3 with Apache 1.3.27, PHP 4.1.2 and MySQL 11.18.

Setting up a web based survey involves many features regarding computers, unrelated to the purpose of the survey and the analysis of the data. The motivation for writing this document was in part to reduce the computer knowledge and skills needed to understand and run WSMS[4]. To run WSMS1.1 we will assume that:

1. WSMS1.1 has access to a server running Apache. That is, in a computer (server) running Apache, there is space (a directory) where the system files (scripts) and people in the Internet

[2]For the Open Source Definition see [7]

[3]While writing this document, Aley, a friend of mine, pointed me to XAMPP([8]), a package with Apache+PHP+MySQL+Perl altogether for both, Windows and Linux. Also, David Solomon sent me the link to Sokkit([9]), a Windows implementation of Apache+PHP+MySQL.

[4]When we found WSMS, we knew Apache only by name and nothing about PHP/MySQL. It took us about 3 months to understand the basics of the software, install it properly and to make WSMS operational. Most of the information was available on the Internet. My first readings were the articles by Kevin Yank at the MySQL web site. Our sincere appreciation to the people sharing their knowledge and work!

can visit the questionnaire and respond to it.

2. The server has PHP installed and running. Note that to run WSMS(1.1) Apache (httpd service) needs the privileges to write the questionnaire in the host directory. If this is a Linux/Unix operation system it is necessary for httpd to be part of WSMS group and all members of the group can write in the directory.

3. MySQL is installed and running. WSMS needs access to a database with the right to create, drop, update and export tables.

3 WSMS Description

The Web–based Survey Mailer System (WSMS) sends out personalized email cover letters while tracking the individuals who have answered the questionnaire so reminder emails are only sent to those individuals who have not done so.

The process begins with emails sent out to the potential respondents containing the link (url) (uniform resource locator, the World Wide Web address) to the questionnaire. Those who respond to it are flagged, so reminder emails will not be sent to them. The responses are captured and saved in the web-server. As a software package, WSMS consists in a set of HTML and PHP scripts which allow you to control who to send the emails to, provide the questionnaire on the web and store the responses.

From the operational viewpoint, WSMS consists in two parts: the *administration side*, run by the WSMS-*user* or *administrator*, who sends out emails; and the *subject side*, the respondent or *client* who fill out the questionnaire and submits it back to the server. For both sides, there are web–forms and information web–pages.

Figure 1 exhibits the administration part of WSMS. Assuming all necessary changes to the code have been made, it starts with a login page [Begin] where you have to provide a password[5]. When you login, a cookie is set and checked for its presence when you access the email cover letter form, so intruders will not have access to the system. If a person other than the administrator accesses the email form, WSMS will check for the cookie[6]. If it is not set, the visitor will be rejected from the form. Only users providing the proper password are allowed to access the form and send out emails. Next the administrator completes a form with the text message of the email to be sent out, including the link to the questionnaire. The system receives the email addresses from the respondents MySQL table in the database, and sends out personalized emails. Once a respondent answers the questionnaire, the system returns to the respondents table and flags the email address so subsequent emails will not reach him or her [confirmation]. The responses are appended to a text file, the responses table.

From the respondent side of the system, the targeted respondent receives an email with the url or link to the questionnaire at the website. The person goes to the web and points the browser to the personalized url provided. For a valid url [Authorized?] the Questionnaire is presented to the respondent and after submission a Thank You Page is displayed. If the url is not valid or the person has already answered the questionnaire, this will not be shown again and an informative

[5] Sans serif font is used to represent the elements of the flow charts in figures 1 and 2.

[6] A cookie is a message the web server sends to the web browser and saved as text file in the client's system. It is used to provide the sever particular client's information. In this case, the cookie is used to check that WSMS1.1 was accessed providing the proper password.

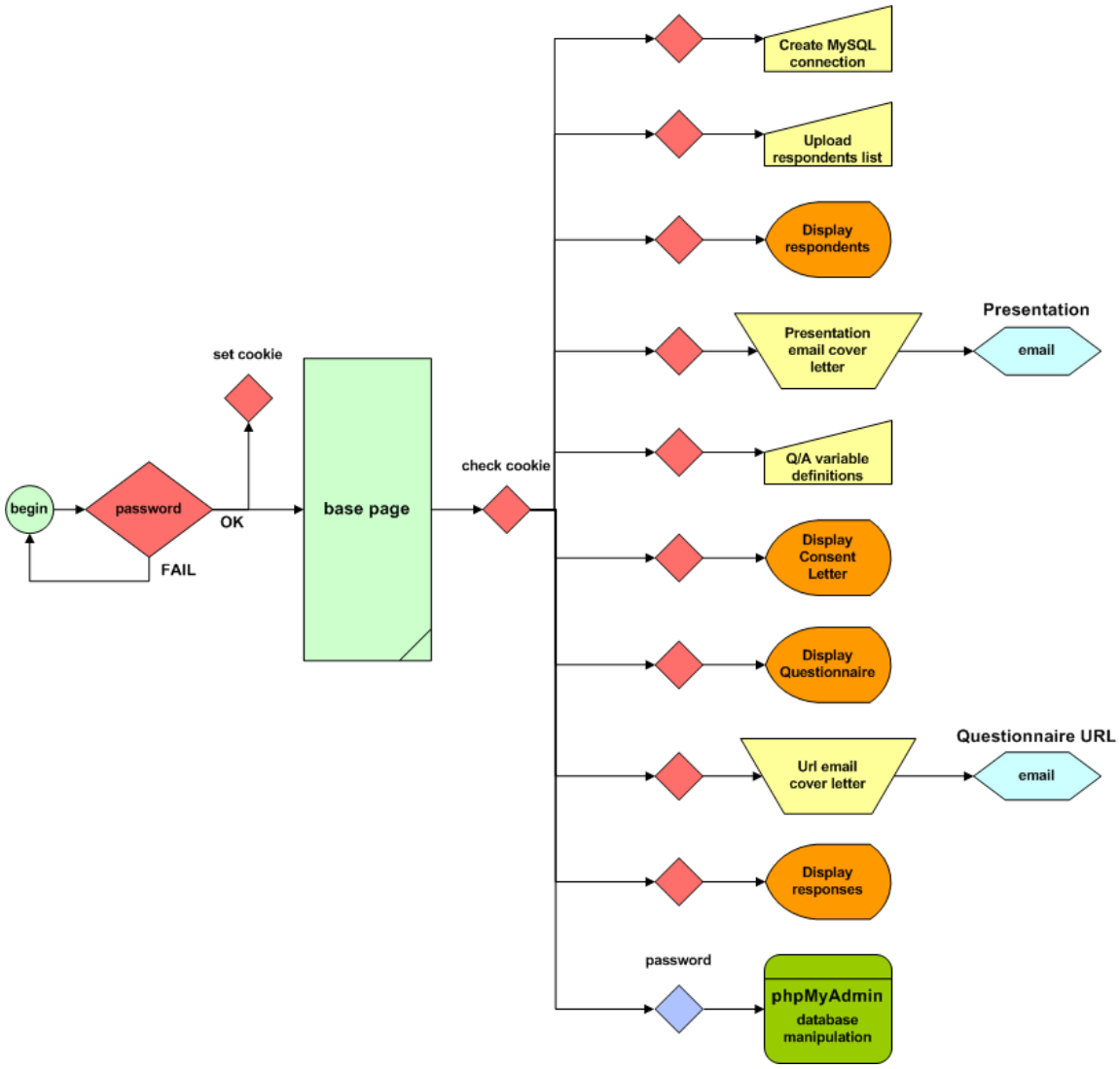

Figure 1: WSMS structure, from the administrator's viewpoint.

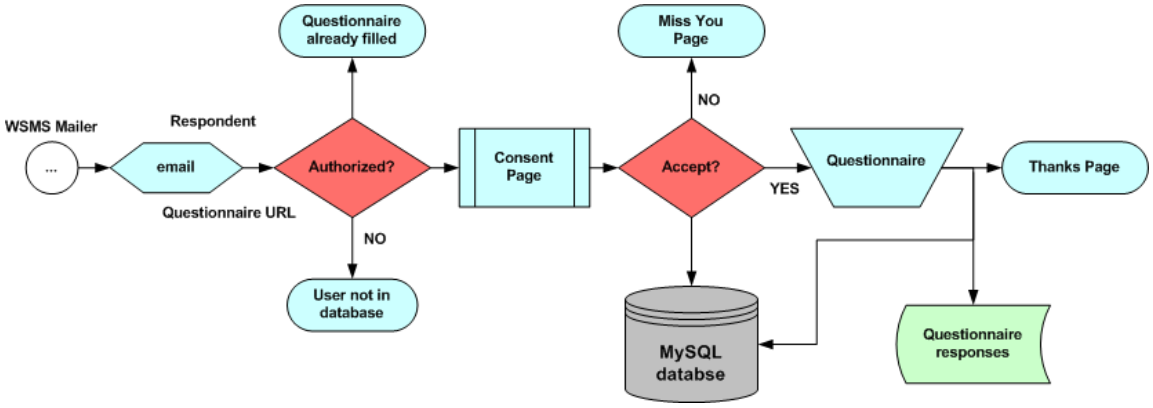

Figure 2: WSMS structure from the respondent viewpoint. MySQL database and the response table are transparent to the respondent.

page ([Questionnaire already filled] or [User not in database]) is displayed. Figure 2 exhibits a flow diagram of the respondent part of WSMS. The [Questionnaire response table] and [MySQL database] elements are transparent to the respondents.

4 WSMS1.1

4.1 What is new?

The first reason for extending WSMS was to reduce the number of scripts of the system that needed to be changed. The advantages are twofold: people less knowledgable on computers can use WSMS easily, and the modular configuration of WSMS1.1 facilitates additions of new features to the system. Another reason was that the experience of actually using WSMS led us to modify some of the scripts for our purposes and to build new scripts for situations that the original system did not cover. Some of the changes and extensions needed were common in all our projects and could be useful in any survey. Figure 3 shows the skeleton of the administration part of the system. For each component in the flow chart, a detailed explanation and screen shots are presented in next subsection. Similarly, figure 4 sketches the respondents side of the system. Details are discussed in subsection 4.3.

The main differences between WSMS and the version 1.1 are:

- After providing the password at the *Login Page* and a cookie set, the survey administrator is taken to the *Base Page* that contains links to all necessary steps (scripts) to carry out the survey. Many of the specifics needed in the process are generated by some of the scripts themselves.

- Via email, the respondents are provided with the url of a *Consent Page* rather than being directly sent to the questionnaire. On the Consent Page, the person is asked to accept or decline to participate in the survey. If the respondent does not accept, she or he will not be sent subsequent emails. See figure 4.

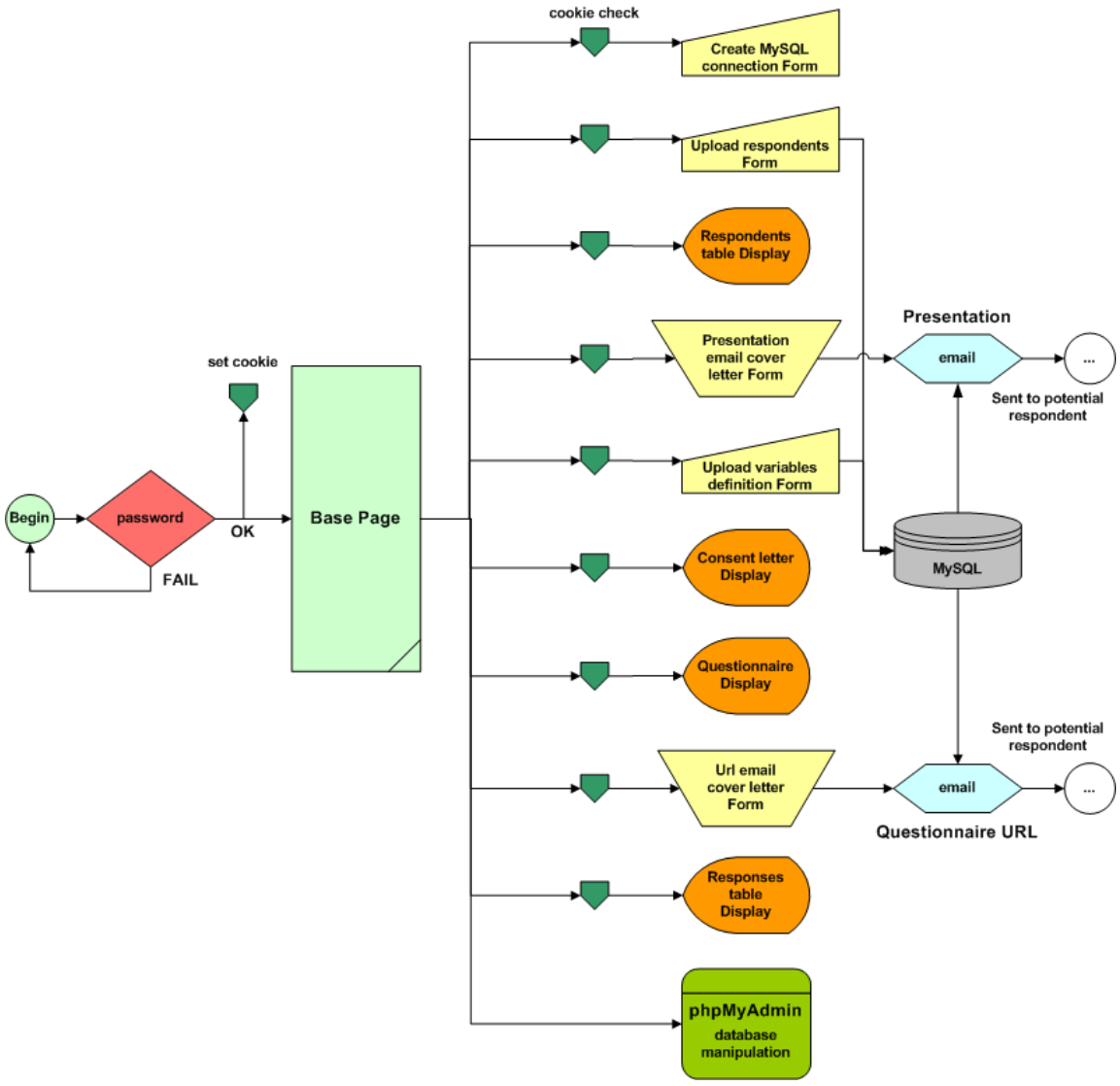

Figure 3: Skeleton of the administration side of WSMS1.1. The Base Page provides, after cookie check, links to each of the Forms and Displays shown in the main body of the diagram.

- The responses are saved in a text file like in WSMS. In addition they are saved in a MySQL table with the advantage that some analysis could be done online using, for example, phpMyAdmin[7]
- We have separated the scripts used only by the administration of WSMS and those presented to the respondents. The last ones are exclusively HTML documents that we call *information pages*. However, when writing the HTML code for the questionnaire, this should be accompanied by a tab delimited file with a list of all the variables used in the questionnaire. This file is used by the WSMS administrator in the *Upload variables form* (shown in figure 10) to create the *responses* MySQL table.

Each element in figures 3 and 4 is self-explanatory. A close look at the displays will give you a quick idea of the system.

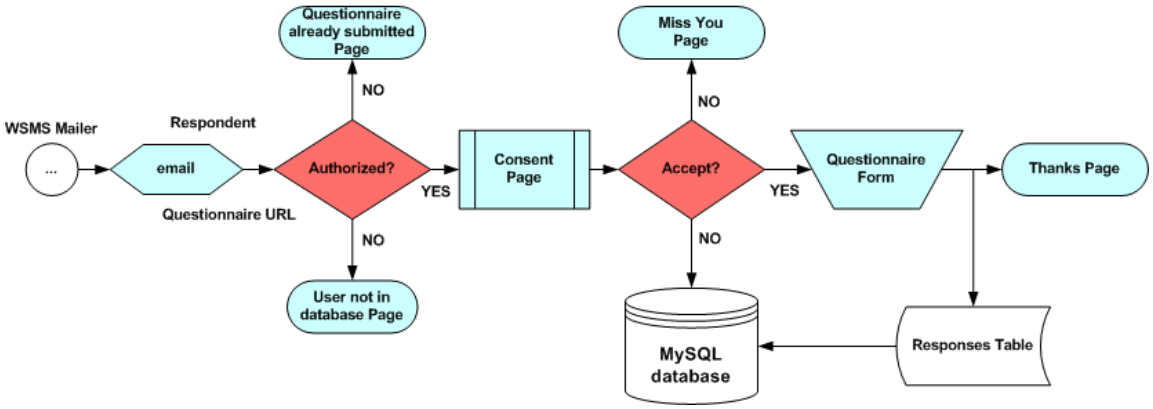

Figure 4: Respondent side of WSMS1.1 structure. Pages and the Questionnaire Form may be displayed. Responses Table and the MySQL database are transparent to the respondent.

In the next section we discuss some details related to the administration of the survey. In section 4.3 we comment on those web pages will be presented to the respondents. To divide the system this way allows to work on the content of the pages, possibly by persons involved in the survey but not necessarily dealing with the administration of the system.

[7] phpMyAdmin is an application to manipulate MySQL tables online from any browser.See [10]

4.2 The administration side.

In this section we discuss the WSMS1.1 administration, run by the *WSMS user* or *administrator*. For sake of explanation, it is assumed that the system has been installed in the server under the *home directory* `/home/wsms/example`, and the necessary changes to the scripts have been made. Also, it is assumed that the pages are accessible on the Internet at the *website* with url http://www.wisc.edu/wsms/example.

To begin setting up the web–survey, first the user has to point the browser to http://www.wisc.edu/wsms/example/login.html to access the *Login Page*, where the WSMS's password is required to get into the system[8]. If the password provided is not correct, the user is asked for the password again. If the password is correct a cookie is set for future checks and the user is taken to the *Base Page*, displayed in figure 5[9]. The *Base page* plays the role of a tasks menu. From there the user may carry out the necessary steps for the survey and visit the *Consent Page* and *Questionnaire* presented to the respondents or *client*. The information pages (HTML scripts) presented to clients can be seen by pointing the browser directly to the scripts. For example, http://wsms.wisc.edu/example/thanks.html will display the page respondents will see after submission of the questionnaire. Note that each job represented in the flow chart of figure 3 has a corresponding entry in the *Base Page*.

WSMS administrator must define the directory, the url and the database location. Next, it is necessary to load the respondents' information to the database and declare the working variables for the questionnaire. The administrator has to send out the emails and waits for the responses. Once in a while the she or he may want to check the responses status to decide when to send out the reminder emails. The *Base Page* provides the links for performing all such tasks.

Except for the information pages, direct access to the any of the WSMS pages will be rejected. The administration of WSMS should go through the the Base Page. Therefore, the user has to login and provide the WSMS password, the purpose of which is to set the cookie. Only authorized persons are allowed to the system. This way, the integrity of the information is protected since it is not possible to tamper with the survey from the information pages.

In the rest of this section we explain each of the links in the *Base Page* and show screen shots of the forms. In next subsection we present the client's side of the system. That is, what the respondent will see: emails, questionnaire and information web–pages. Some details are discussed in section 5.

[8] This password can be anything you like and is checked against to what you defined in the `logcheck.php` script. We used the same password, `wsms-passwd`, for WSMS1.1 and MySQL but they could be different.

[9] Writing this document I used the Internet browser K-Meleon to follow up the procedure([11]). Some other and definitively more popular browsers are Netscape and Internet Explorer.

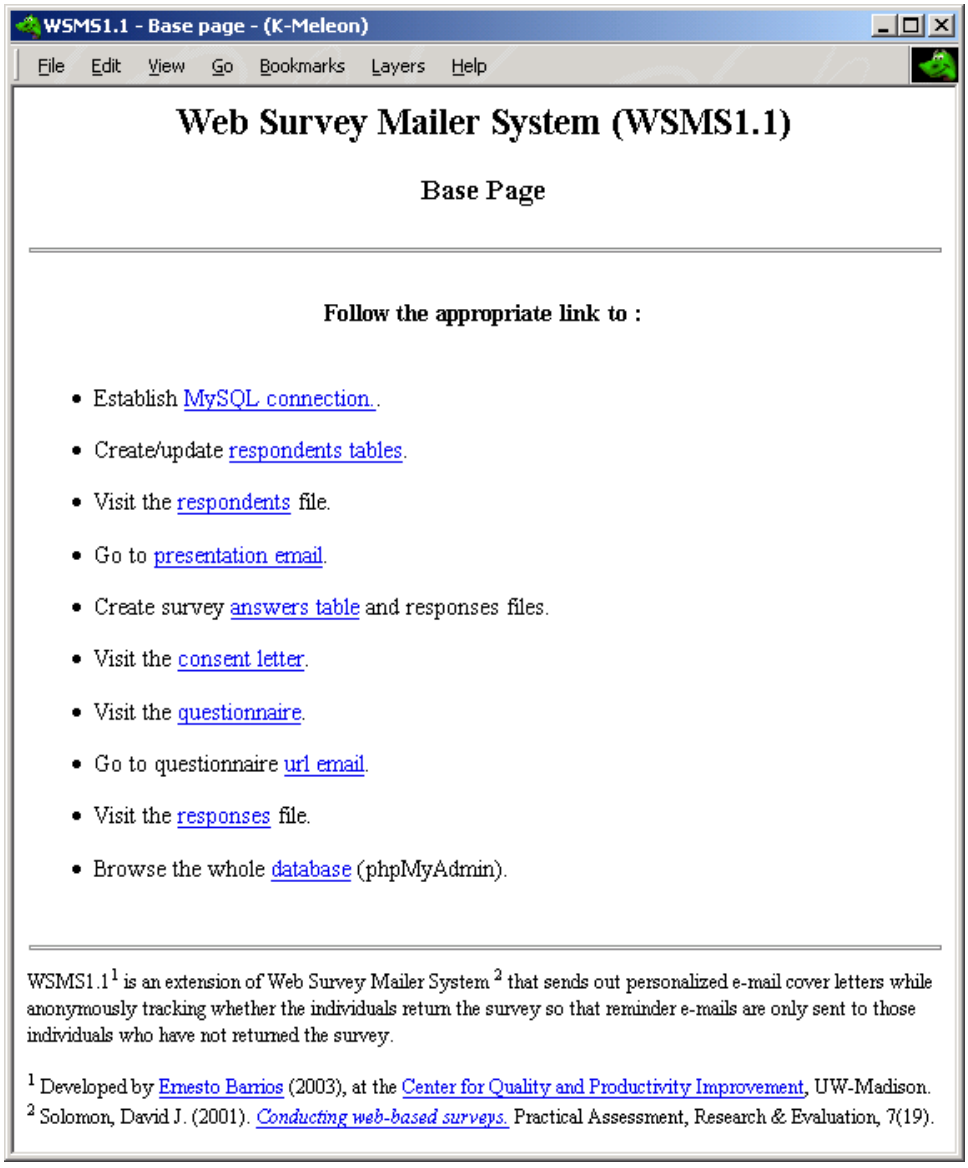

Figure 5: Base Page.

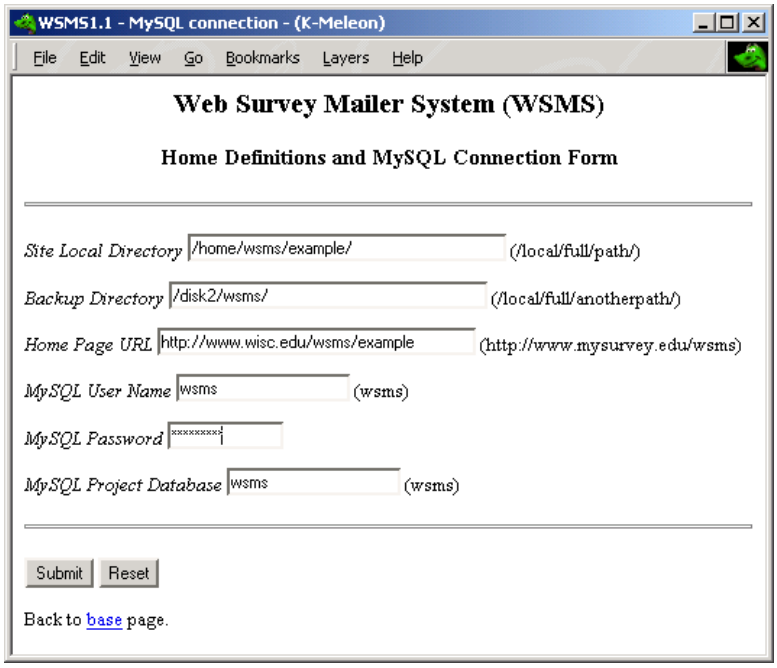

Figure 6: Home definitions and MySQL Connections Form.

Establish MySQL connection [Figure 6] This is the first link in the *Base page*. In this form we declare where the files are located in the server and where on the Internet the pages can be found. The different entries are:
◇ Site Local Directory: Provides the location of the WSMS files for the survey are on the server (absolute path), assuming the files are in the directory `/home/wsms/example/`. In Windows this path could be something like `c:/wsms/example/`[10]. Note that you have to finish the path with the forward slash `/`.
◇ Backup Directory: Under this directory a copy of the responses file is saved with extension `bak`. If left empty, the home directory is used.
◇ Home Page URL: WSMS's web address (url). You have to include the protocol `http://`.
◇ MySQL User Name: WSMS will use this user name to access MySQL database manager. In this example, wsms itself is a valid user.
◇ MySQL Password:. The corresponding password. In our example, `wsms-passwd` was used.
◇ MySQL Project Database. The database WSMS will save the respondents and responses tables.

After submission you get a message and a link to go back to the *Base Page*. The system saved the data just provided in the file `connect.inc` and this data is used in various stages of the process.

The next step is to generate the respondents MySQL table. This table has the name and email address of the potential respondents. The table is used whenever an email is sent out. It is also used to record whether a person accepts or rejects participation in the survey and if he/she has filled out the questionnaire, so reminder emails are not sent to him/her.

[10]Under Windows 98 you may need to replace the forward slashes "/" by backslashes "\".

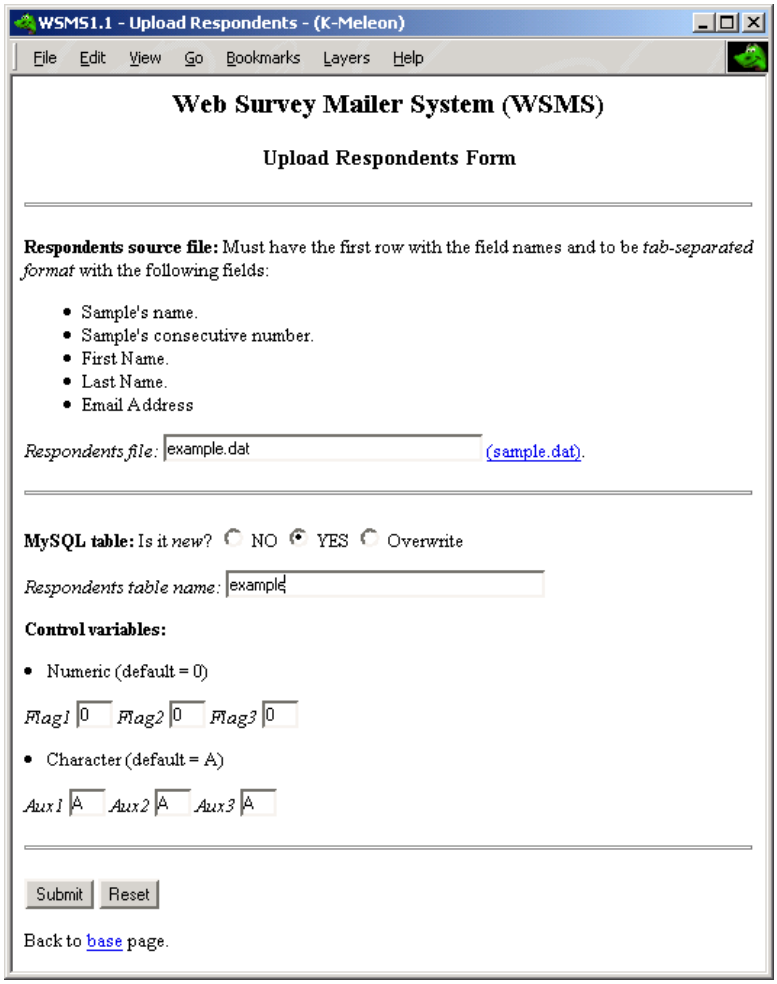

Figure 7: Upload respondents form.

Create/update respondents tables [Figure 7] This form creates/updates the MySQL table and a text file with the respondents data.
◇ Respondents file: This should be a tab delimited file with the respondents' information[11]. The first row of the file is ignored and may be used for the name of the *fields* (columns). The columns correspond to the name of the population (group) from which the survey is taken; the consecutive number; the first and last name, and the email address of the respondents. See section 5 for some details.
◇ MySQL table. Is it new?: Click NO if the table already exists and you want to append a new list of respondents; YES, if you are creating the table, otherwise Overwrite for replacing existing data.

[11] You can create a tab delimited file with any editor simply using the [tab] key to separate columns. If a first name has two words, like "Johann Sebastian", use the [space] key between the names. An easy way to build the type of formatted file is using a spreadsheet application like OpenOffice Calc, MS Excel, or Lotus, and the *Save As* option.

◇ Respondents table name: How you want to name the respondents MySQL table.
◇ Control Variables: These are used to define whom the emails are sent to in the *Presentation Email Form* and the *URL Email Form*, shown in figures 9 and 13 respectively. Currently, only the numeric variables *Flags* are used. The character variables *Aux* are created and stored anyway. The default values of the variables, 0 (zero) and A, are displayed in the form.

After the form submission a respondents MySQL table is created/updated and the new data uploaded. A new column *id* is appended in the table. This *id* is unique to each respondent and used for questionnaire responses. Additionally a PHP script (`respondents.link.php`) is generated with the link to the text file with the respondents table status. The file name is created appending `.out.dat` to the name of the table. name. Administrators receive an acknowledgement message and the file uploaded is displayed.

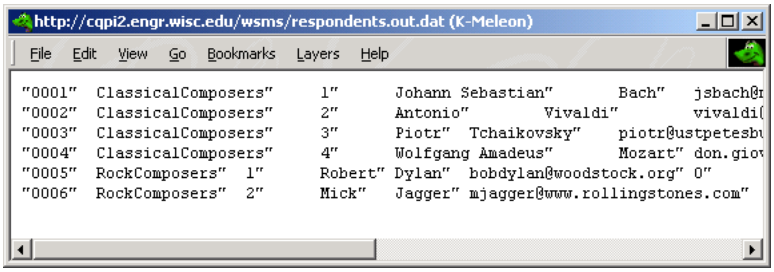

Figure 8: Respondents file example.

Visit the respondents file [Figure 8] The respondents file is updated whenever the link is followed. The current respondents MySQL table is read and exported to the text file displayed. Use it if you want to monitor the *Flags* status. Point your browser to the respondents file, in our example `respondents.out.dat`, if you want to download the respondents data.

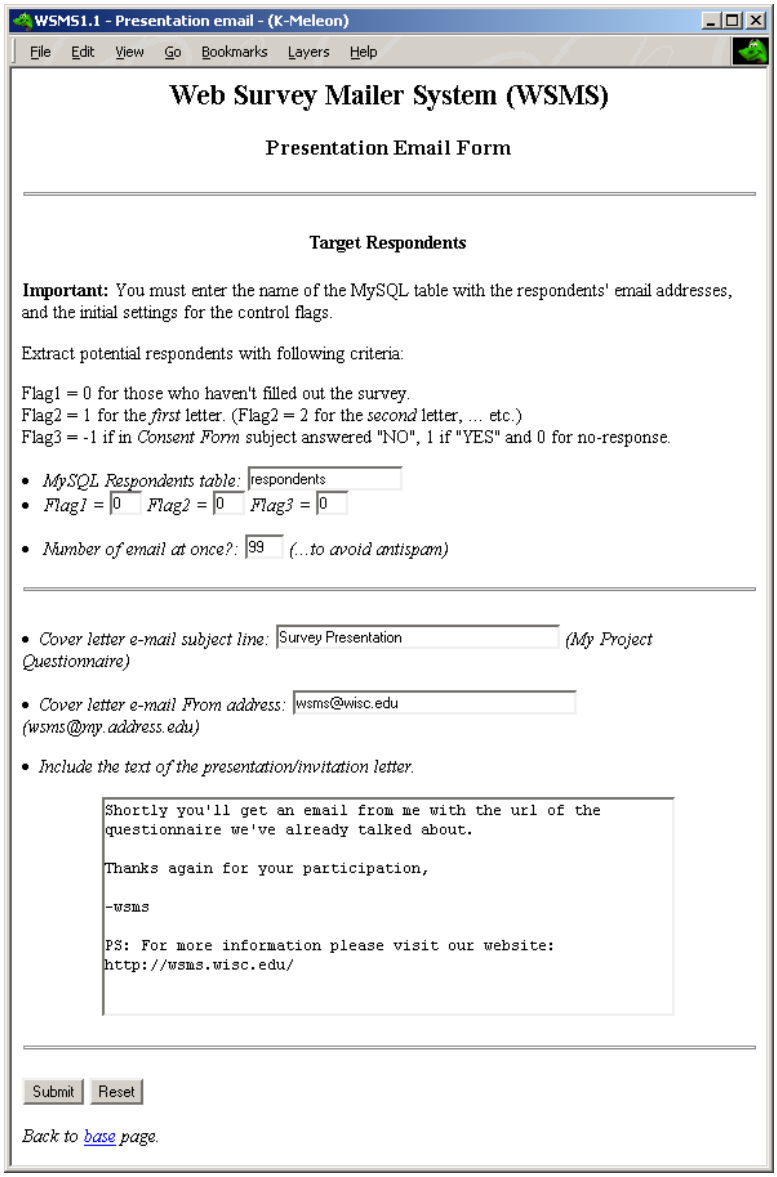

Figure 9: Presentation email form.

Go to presentation email [Figure 9] This form allows you to send personalized emails to the individuals in the respondents MySQL table. It was designed to reach the respondents regardless the Consent Letter or the Questionnaire. It could be used, to introduce yourself or your project.
◇ MySQL Respondents table: The name of the MySQL respondents table.
◇ Flag 1–3: *Flag*s values of the respondents you want to send the email. Rules of how *Flag*s change are described in section 5.

◇ Number of emails at once: Maximum number of emails you want to send out per batch. This feature is useful when you have a limit time for individual PHP process, or to avoid anti-spam filters for the amount of emails. If for example, you want to send 220 emails but you have the limit of 50 per batch, set the number to 50, and submit. You get an acknowledge page displaying the number of emails just sent. (In the background the corresponding *Flag2* has been increased by 1.) Use the [⇐ Back] feature of your browser and submit the form again. Repeat the process until all the emails are sent.
◇ Cover letter email subject line: The text you want in the *Subject* field of your email.
◇ Cover letter email From address: The email address you want in the *From* field of your email[12].
◇ Email text body: Insert the text of the email. The emails are personalized, so the salutation line will be, for example, *Dear Mick*, where *Dear* was defined in the `presentation.email.php` file and *Mick* was taken from the respondents MySQL table.

After submission you get an acknowledge page with the number of emails sent. Figure 16 displays how the example email would be seen by the recipient.

[12]In Solomon's WSMS, the *Return-path* was used instead of the *From* field. This is useful when you have emails that did not reach the recipient and bounced back. However, in all the different systems WSMS1.1 was tested this feature was not permitted and undelivered emails bounced back to Apache. You may need to contact your Apache administrator to let you know about bounced emails.

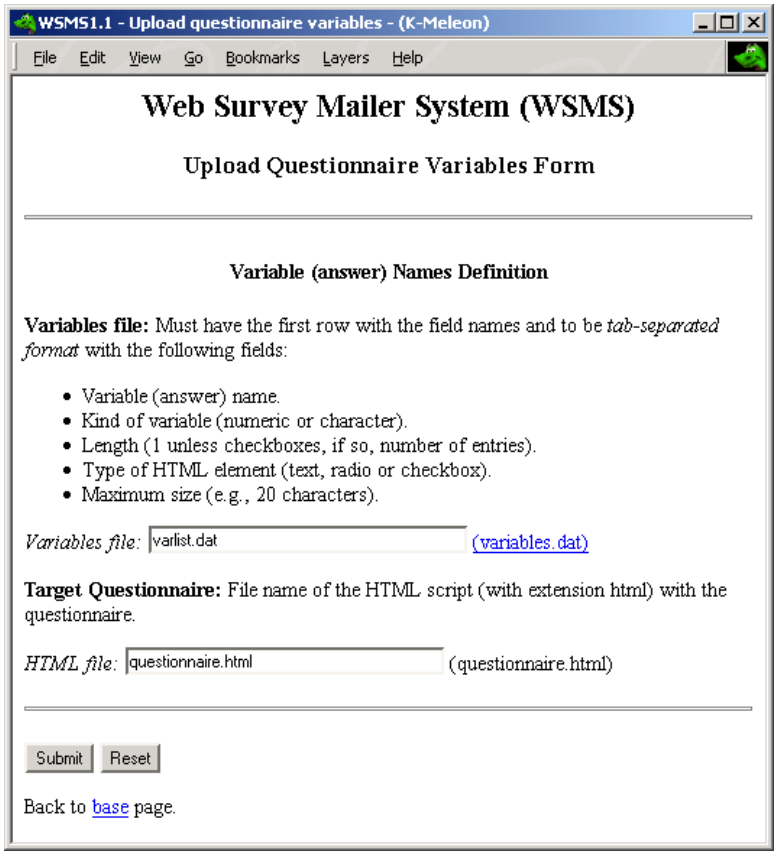

Figure 10: Upload variables form.

Upload Questionnaire Variables Form [Figure 10] This step creates the *responses* MySQL table and text files.

◇ Variables file: Provide the name of the tab delimited text file with the name of the variables used in the questionnaire. This file has to include 5 fields per line: variable name; kind of variable (numeric or character); length (1 unless the variable is an array, in that case the corresponding number of entries); type of HTML element (text, radio button, checkbox or select for multiple selection menus); and maximum size (the longest string expected when the kind of variable is character). The first line is dismissed and can be used for column names. See section 5 for more details.

◇ HTML file: The name of the HTML script with the questionnaire. This is used for naming the generated PHP scripts. More details are discussed in section 5.

After submission a MySQL table is created. This is the table which will host the responses to the questionnaire. Two files are created for the text file version of the table. A list of all the created variables is displayed.

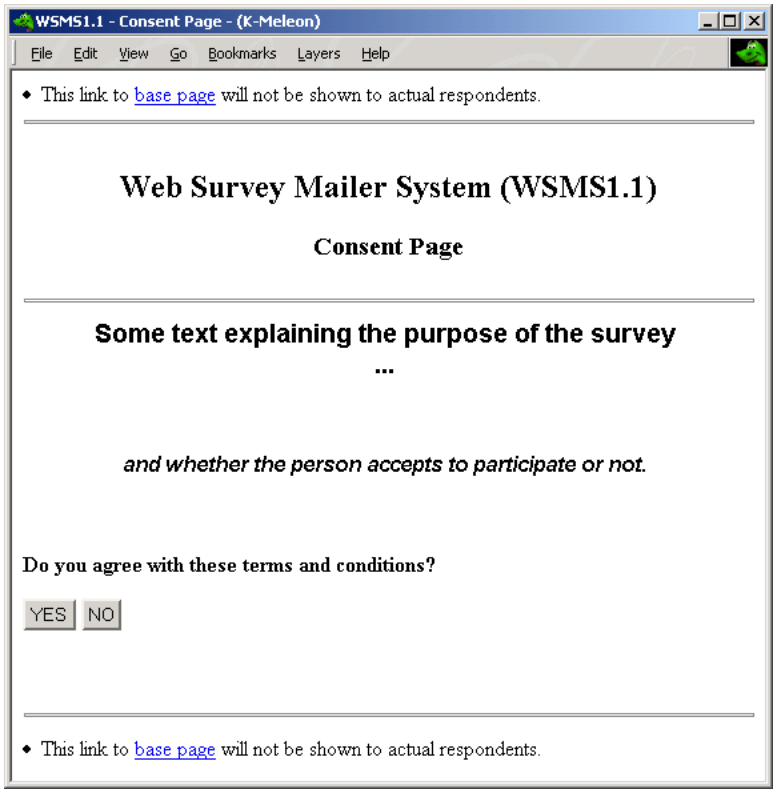

Figure 11: Consent form example.

Visit the consent letter [Figure 11] Following the link will display how the *Consent Letter* will be seen by the respondents. The display includes links to the *Base Page* which are not shown to the respondent. Clicking on the any of the buttons will take you to the corresponding web page the respondent would see. We comment on this and all other HTML scripts in the next section.

Visit the questionnaire [Figure 12 (p.21)] The questionnaire is displayed with links to the *base Page*. In this case, as administrator and coming from the *Base Page*, submission of the form will not have any effect on the response table. Comments on the construction of the HTML script for the Consent Letter and the Questionnaire are included in the next two sections.

WSMS1.1 package includes a questionnaire example showing the use of different HTML elements: *text box*, *radio buttons*, *check boxes* and *menus*. See section 5 for some details on some HTML elements.

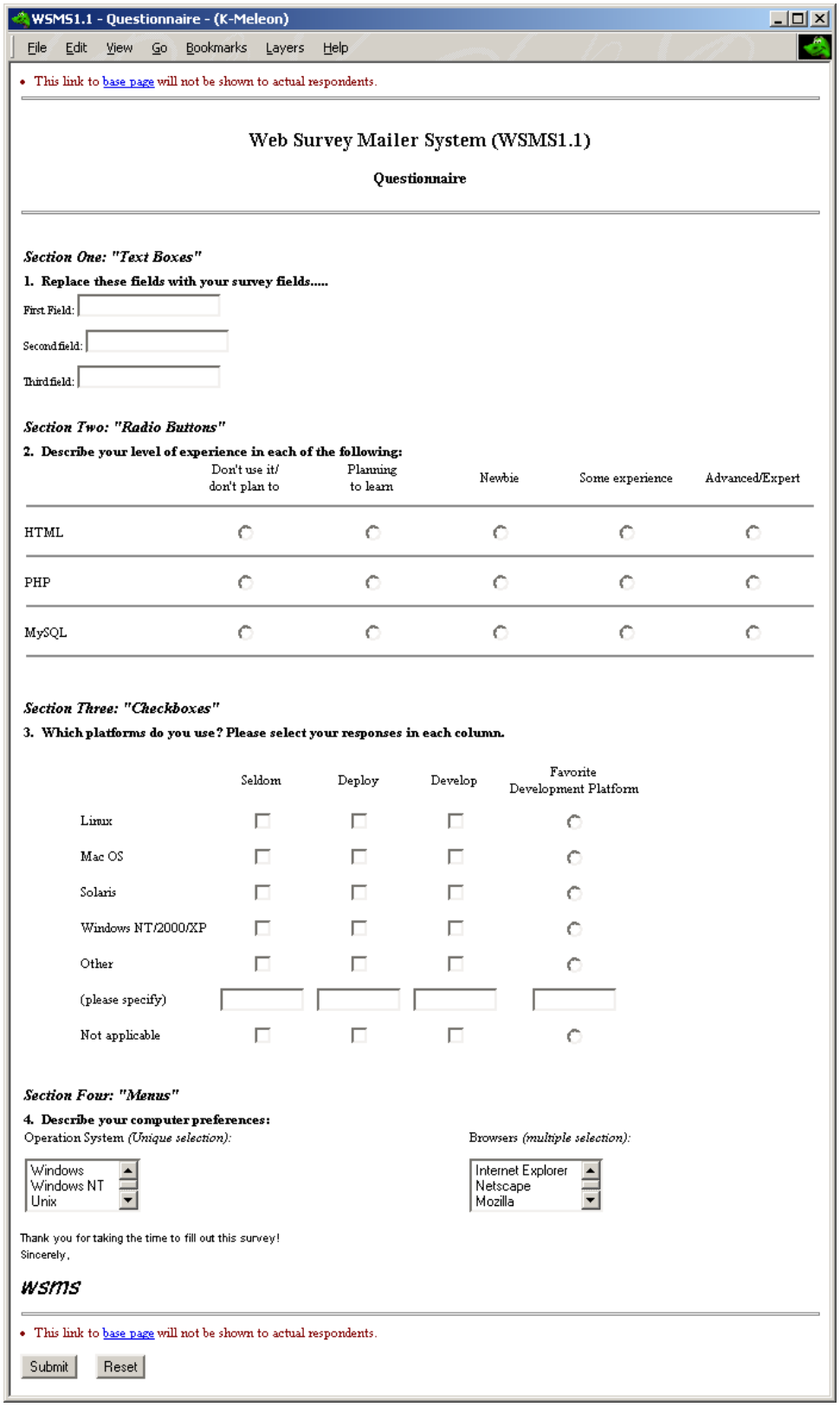

Figure 12: Questionnaire. Text boxes and radio buttons example.

Go to questionnaire url email [13] [Figure 13 (p.23)] The main differences between this email form and the *Presentation Email Form* (figure 9) are: a) you decide whether the survey is completely anonymous or the person's *id* is recorded along with his/her responses, and b) you provide the url to your survey.

◇ Anonymous response: If NO the respondent's *id* is recorded. Otherwise, `xxx` is used as identification.

◇ URL of the consent form: Provide the url of the *Consent Letter*. The body of the email is split in two by a link to the url[14]. Figure 17 exhibits how the recipient would receive this kind of email.

[13] This form is very similar to the original WSMS. The additions in the 1.1 version are: 3 control variables (*Flags*), the maximum number of emails and the option for anonymous tracking.

[14] If the Consent Letter is not used in your project you may provide instead the Questionnaire's url.

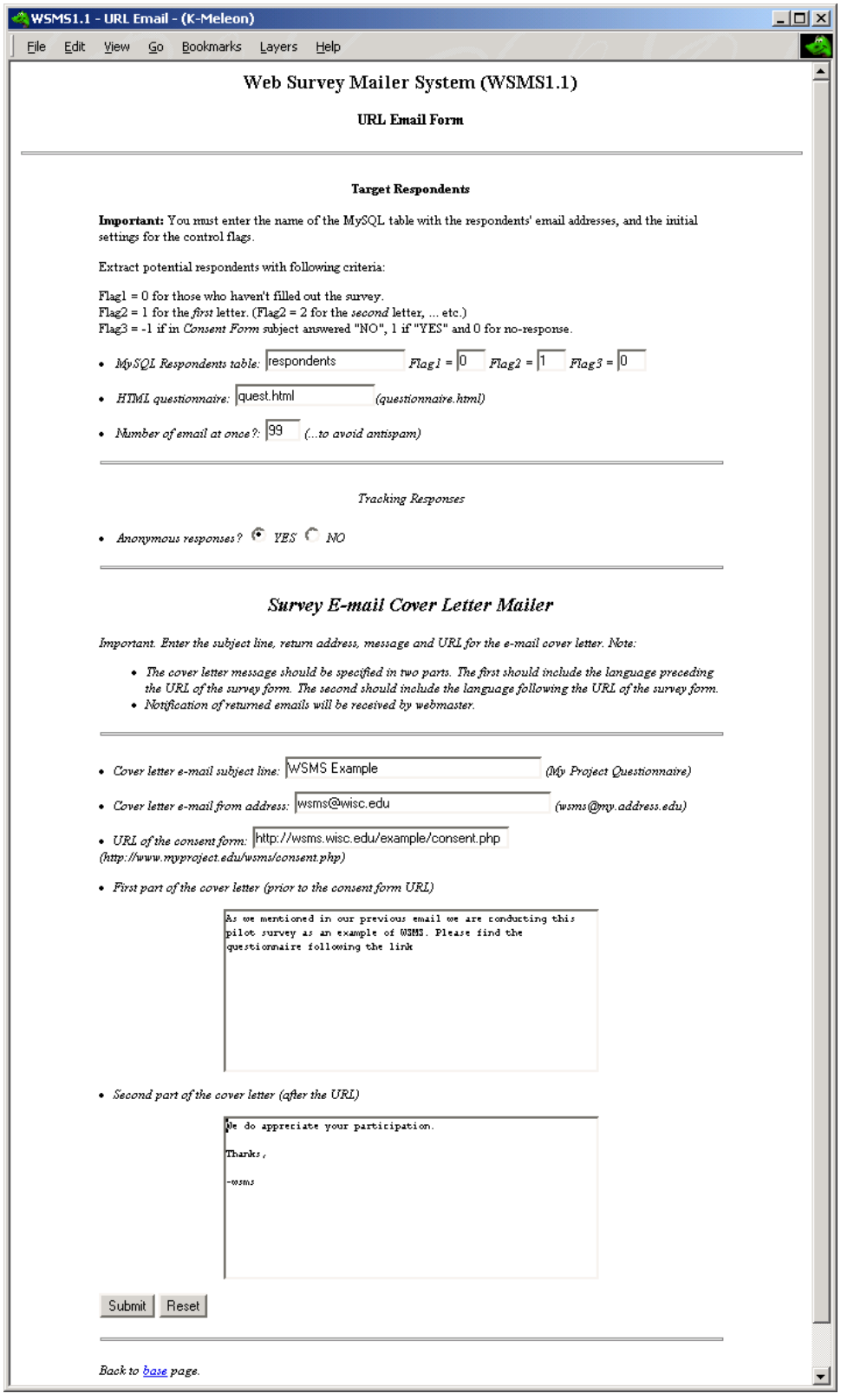

Figure 13: Url email form.

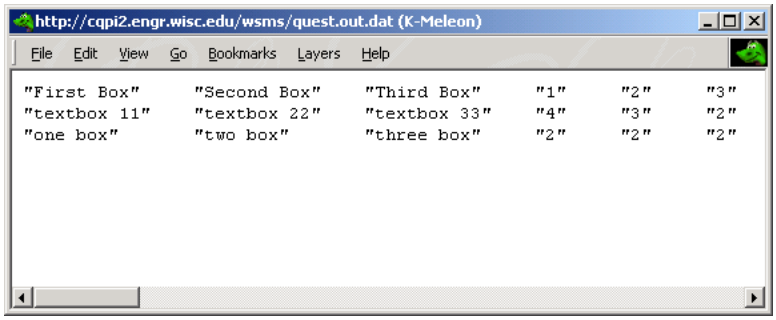

Figure 14: Example of the responses file.

Visit the responses file [Figure 14] The responses file is displayed. Like the respondents file, each question–response (field) is quoted and each line corresponds to one person's questionnaire response. You can point your browser to the file to download it. In our example, this would be http://wsms.wisc.edu/example/quest.out.dat.

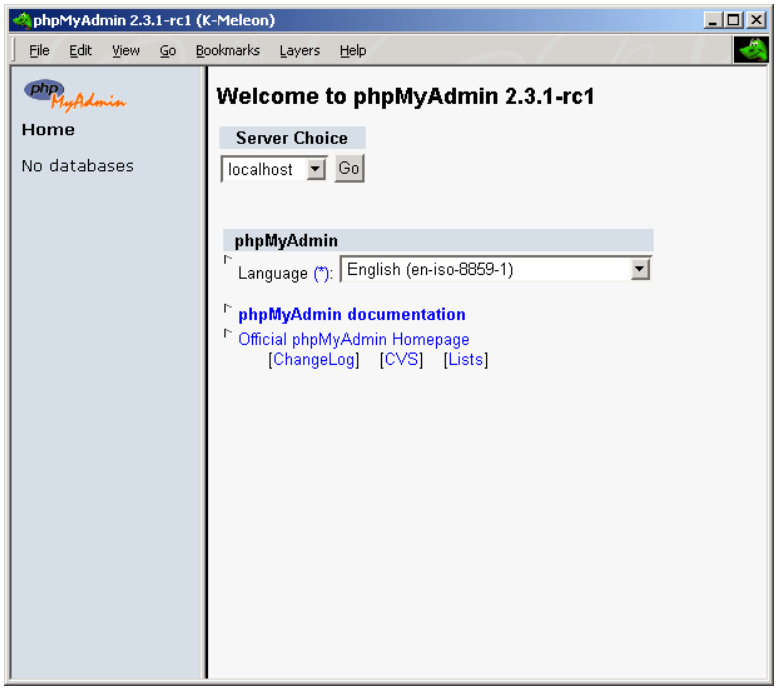

Figure 15: Front page of phpMyAdmin.

Browse the whole database (phpMyAdmin)[Figure 15] This link is optional. phpMyAdmin *is intended to handle the adminstration of MySQL over the web.* Briefly, it is a package of PHP scripts that you have to install as another site in your web server and registered as MySQL user with

some particular privileges. As mentioned already, it allows you to administer MySQL databases from any browser over the internet. In particular, it is handy to have it for manipulating MySQL tables in cases not considered by the WSMS1.1. For example, changing some email addresses when the respondents table has been uploaded. If phpMyAdmin is not available in your server you can always manipulate MySQL tables from a shell in a command session[15].

4.3 The respondent side

One of the most important parts of a survey is the design of the questionnaire. As we indicated in the introduction WSMS is a *tool* for carrying out the survey over the web, however does not address the design of the survey or the questionnaire. This section briefly comments on the pages the respondents may visit.

As mentioned before, an issue considered in the design of WSMS1.1 is the separation of the administration duties and the pages and forms presented to the respondents. This approach has the advantage of separating the tasks involved in a survey, setup and questionnaire, so that two or more persons may work on the survey without interfering with each other. For example, it could be the case that surveys are simultaneously run. In this situation, one person as administrator, could setup the different surveys while others may be working on their own pages and questionnaires (HTML scripts).

As far as the potential respondents are concerned, they will receive one or two emails: a presentation email, where your project is introduced, and an email with the url of the Consent Page. These emails, sent using the *Presentation Form* and the *Url Email Form* (see figures 9, 13) will look as those shown in figures 16 and 17 respectively.

The respondent, following the url–link provided in the second email will be presented with a *Consent Page* like the one shown in figure 11. The body of the page is HTML coded in the `consent.html` file. The links to the *Base Page* at the top and bottom of the page are declared in `consent.php`. You may want to use this *Consent Page* to present some information relevant to your project and include some links related, like your project's website or the email addresses of the project leader or the survey webmaster.

If the person does not agree to participate in the survey, ([NO]), the *Miss you Page*, like the one displayed in figure 18, is shown. Again, you may want to include some information about your project and indicate that the respondent can always change his/her mind, go back to the *Consent Page* and click on [YES] this time. If, on the other hand, the person agrees to participate in the survey, he/she is taken to the questionnaire page. Figure 12 exhibits the questionnaire example in WSMS1.1. The questionnaire includes the most common forms of getting information from a visitor by a website. We discuss some details in the next section.

The questionnaire has the [Submit] and the [Reset] buttons at the bottom of the page. The [Reset] button will clear the answers already on the page. That is, the responses will be deleted and the page refreshed. You may want to warn the respondent of this feature, particularly if it is a long questionnaire. Few people will fill out the form again if the questionnaire takes 15 minutes or more to complete. If the person submits the questionnaire, he or she is acknowledged with a *Thank you Page*, like the one in figure 19.

[15] There was the Windows application MySQL-Front but its development was discontinued by 2002. The graphical interface myFrontend started last October but I do not have any experience with it. The development of the graphical sql client for MySQL, MySQLGUI, was stopped and being replaced by MySQL Control Center, both from the MySQL initiative.

```
Date: Fri, 6 Dec 2002 14:31:56 -0600
To: mjagger@www.rollingstones.com
Subject: Survey Participation
From: wsms@wisc.edu
Reply-To:wsms@wisc.edu
X-Mailer: PHP/4.1.2

Dear Mick,

Shortly you'll get an email from me with the url of the questionnaire we've
already talked about.

Thanks again for your participation,

-wsms

PS: For more information please visit our website: http://wsms.wisc.edu/
```

Figure 16: Presentation email text.

```
Date: Fri, 6 Dec 2002 14:41:06 -0600
To: mjagger@www.rollingstones.com
Subject: WSMS Example
From: wsms@wisc.edu
Reply-To:wsms@wisc.edu
X-Mailer: PHP/4.1.2

Dear Mick,

As we mentioned in our previous email we are conducting this pilot survey as
an example of WSMS. Please find the questionnaire following the link

http://wsms.wisc.edu/example/consent.php?resp=example&surv=quest&key=0cker7

We do appreciate your participation.

Thanks,

-wsms
```

Figure 17: Url email text.

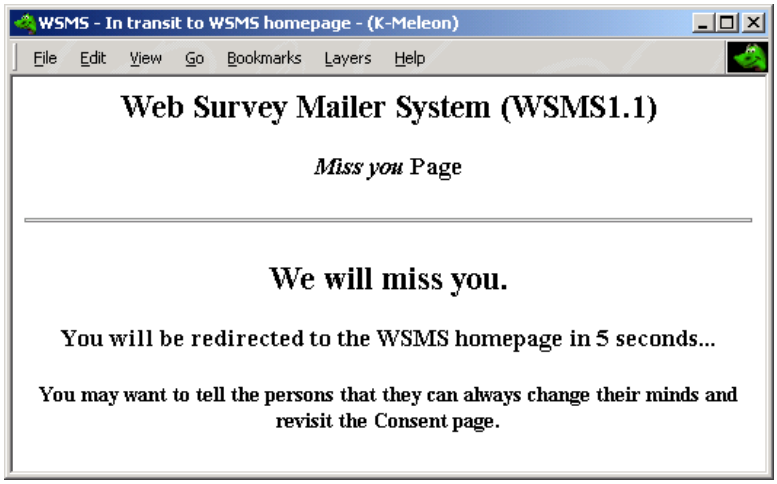

Figure 18: Miss you page.

Finally, there are two more pages the respondent may see. The *Already Page* and the *Wrong ID Page*. See figures 20 and 21. The former is shown when a person submits the questionnaire a second time. The later when the url of the *Consent Page* is correct but the keyword is wrong. Only authorized respondents can submit questionnaires.

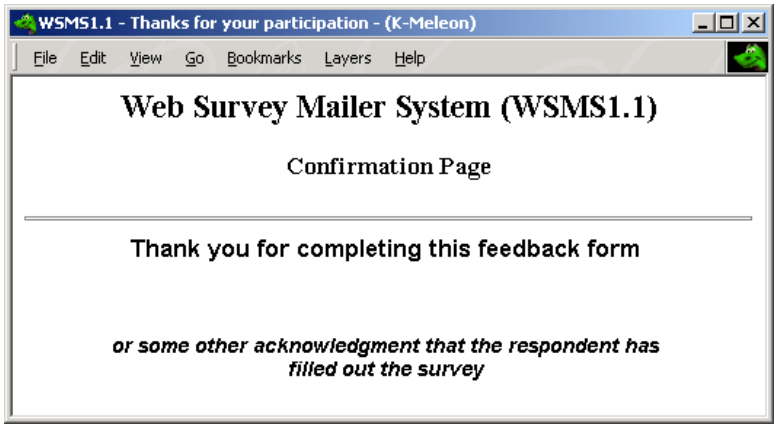

Figure 19: Thank you page.

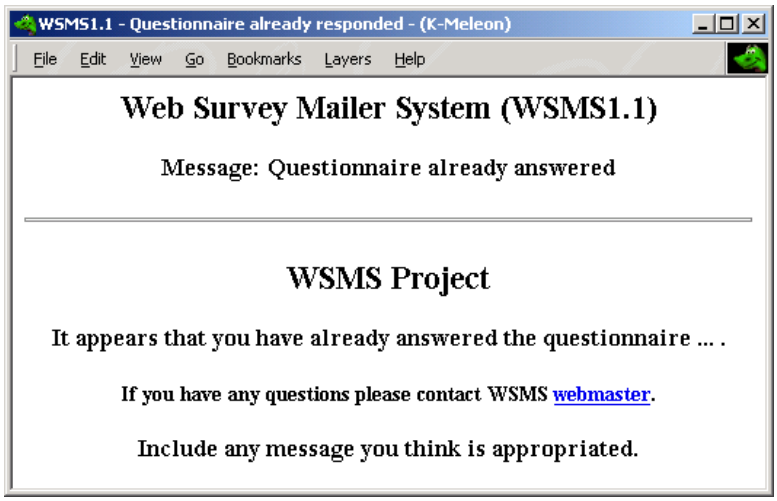

Figure 20: Questionnaire already responded message.

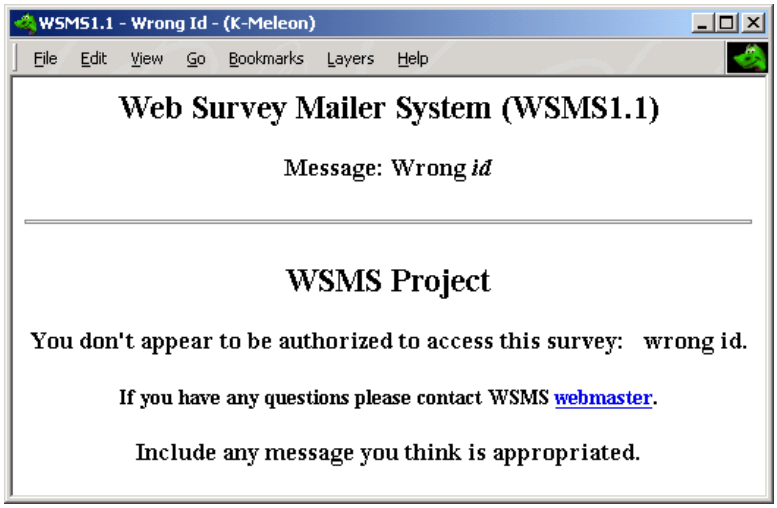

Figure 21: Wrong keyword message.

5 Some Details

As mentioned before, it is assumed that the `httpd` service (Apache) has the permission to write in the home directory `/home/wsms/example`. For security reasons, it is *strongly* recommended that the httpd settings for that particular directory are set so the list of files is not displayed under http://wsms.wisc.edu/example/, since name and email address of the respondents are saved in some files in the same directory. These settings are defined in the hidden file `.htaccess`. Alternatively, many servers are set so `index.html` is shown by default. WSMS1.1 includes an example of such a file.

To setup WSMS 1.1, it is necessary to modify the file `logcheck.php` to consider the password you want for access to WSMS1.1. This feature prevents unsolicited visitors from accessing the different forms. This password may be different to the one used for MySQL. Once logged into the system a cookie is placed on the system and will be checked whenever you visit a WSMS page. Then the *Base Page* is displayed.

After submission of the *Connection Page* the file `connect.inc` is created with the information just provided. A list of all the files in the WSMS1.1 package is included as appendix A.1 with a brief explanation for each of them. We indicate which files are generated automatically and which others need to be changed by the administrator.

When uploading the respondents information in the *Respondents Form* (figure 7) nake sure that the respondents file is tab delimited. This type of file format is easily created with a spreadsheet application. The file has to have the following fields or columns: population (sample) name, unique (consecutive) number, first name, last name and email address. The first row of the file is ignored and can be used for column names. See for example `respondents.dat` file in WSMS1.1 package.

There are two control sets of variables declared in the *Respondents Page*. In this version 1.1 only *Flags* are considered. They are numeric variables and used to identify whom to send the emails. The three variables are set to 0 by default but any value could be used. They change accordingly to the following rules:

Flag1:	Changes to 1 when the person submits the questionnaire.
Flag2:	Increments in 1 for each email the person receives.
Flag3:	Changes when the *Consent Page* is submitted. To 1 (-1) if the person does (not) accept to participate in the survey.

The respondents MySQL table is created or updated after submission. The option NO for creating a new MySQL table is used when you want to append new respondents to a table already in your database. The respondents table has the following fields:

Num	Consecutive number in the whole table. Generated by WSMS1.1
Sample	Name of the population or sample defined in the respondents file.
SampleNum	Person's number defined in the file within the sample.
FirstName	Defined in the file.
LastName	Defined in the file.
Email	Email address. Defined in the file.
Flag1–3	Numeric control variables. Default value 0.
Aux1–3	Character control variables. Defaul value A. Currently unused.
Id	Unique identification key. Generated from the last 2 characters of the first name, the first 2 characters of last name and 3 times table consecutive number plus 1.

The keyword is attached to the url in the email sent to the respondents so WSMS1.1 will identify the person and update accordingly the MySQL tables. At this moment the file `respondents.link.php` is created and will be used linking the *Base Page* with the respondents file `respondents.dat` that is updated whenever you visit it. For a copy of the current respondents file, either, highlight *all* the page, copy and page into a text editor, or you can use phpMyAdmin to export the MySQL table. Using the *Save As* or *View Source* options of your browser will include the HTML tags used for the display.

Any time an email is sent out, you have to identify the potential respondents by means of the *Flags*' values in the *Presentation Email Form*. Also, you have to declare the maximum numbers of emails sent per batch. After an email is sent out, the corresponding *Flag2* is incremented by 1 and the total number of emails counter's advanced. If for any reason the batch could not be finished, only those emails actually sent are registered. You have to submit as many batches as necessary.

Note the characters `?resp=example&surv=quest&key=0cker7` after the page's url in the email shown in figure 17. This string informs WSMS1.1 which MySQL respondents table has to be updated (`example`), the name of the questionnaire form (`quest`) the person is answering and the respondent's *id* (`cker7`). Also, note that the emails are personalized. Both, `Mick` and `mjagger@www.rollingstones.com` were taken from table respondents MySQL table.

When the link to the questionnaire is followed from the *Base Page* or what the respondents sees is the HTML script of the questionnaire referred to in `quest.php`. This PHP code calls the scripts `quest.head.inc` and `quest.bottom.inc` that include links and variables necessary for the questionnaire. The questionnaire is written in HTML. The questionnaire example in WSMS1.1 package is `quest.html`.

The questionnaire example considers the most common forms used when people are asked for information in the Internet. For example, your age, professional area, income range, your opinion on a certain topic, etc. To capture the information, you may be presented with *boxes* where you can type your response, (say, first name); *radio buttons* (how many persons live in your household: 1, 2, 3, or more); *checkboxes* (type of music you listen more often: blues, jazz and hard rock). Sometimes the possible responses are presented as pull-down menus. See for example figure 12.

In addition to the effort employed in the design of the questionnaire, writing the HTML code will take most of your time when setting up the web survey. This is true if you are not familiar with HTML or if your questionnaire is very long. Nevertheless we have to emphasize that you need not be an expert on HTML to write a nice questionnaire. Just keep in mind that the questionnaire is part of a system and has to communicate with other scripts. In particular, when a question is asked, the response should be perfectly recognized by the system and this is done by a variable name.

In the process of writing the questionnaire code it is recommended to create the tab delimited variables file (`varlist.dat`). This file is used to generate the responses MySQL table. Each variable in the questionnaire corresponds to one row in the file. Each row has the following columns:

name:	Name of the variable. Will be used by some PHP scripts.
kind:	Character or numeric. Used to define the variables in MySQL.
length:	Numeric 1 unless it is an array. In that case the number of entries in the array.
type:	HTML element: *text*, *radio button*, *checkbox*, or *select* (menus).
size:	Maximum number of character or digits.

As in the respondents file, the first row is skipped and may be used for column headers. The *Upload Variables Form* calls the file `answers.table.php` which creates the responses table and the text

files where the answers will be saved. All the variables in the MySQL table are taken from the variables file. Those declared as character and numeric type are defined in MySQL as TINYTEXT and TINYINT respectively. The last 3 variables in the table: *date*, *time* and *id*, are generated with the questionnaire submission. If the survey is run as *anonymous*, `xxx` is recorded instead of the actual *id*.

For a large MySQL variables table you may want to edit the `answers.table.php` file to include more options in variable declaration. The same script will also generate two PHP scripts, `survey.inc` and `survey.link.php` in our example. The first script joins all the individual variables in an unique array used later when collecting the responses. The second one generates a link to the responses text file.

The table and files names are defined from the name of questionnaire file name. For example, if the questionnaire HTML script is `quest.html`, MySQL responses table's name is `quest` and the text files are `quest.out.dat` and `survey.out.bak`. The backup file is saved in the directory indicated in the *Connection Page*.

For example, in section 2 of our questionnaire, *Radio Buttons*, the first question, has 5 possible responses presented as radio buttons. This question is coded as:

```
<TD>HTML</TD>
    <TD align=middle><INPUT type=radio value="1" name="radio1"> </TD>
    <TD align=middle><INPUT type=radio value="2" name="radio1"> </TD>
    <TD align=middle><INPUT type=radio value="3" name="radio1"> </TD>
    <TD align=middle><INPUT type=radio value="4" name="radio1"> </TD>
    <TD align=middle><INPUT type=radio value="5" name="radio1"> </TD>
```

Then the response variable `radio1`, will take values 1, 2, 3, 4 or 5, depending on which button was clicked. Accordingly, PHP scripts make reference to it (or its value) as `$radio1`. The the variables file would have the row:

```
VarName VarKind Length  Type    Size
...
radio1  numeric 1       radio   1
...
```

When a checkbox or a multiple select menu makes reference to all the possible options with a unique variable name, WSMS1.1 uses the declared *length* and creates the necessary variables so that each option can be referred individually. The variables just created will have the original variable name with `_1`, `_2`, etc., appended. See the generated file `quest.var.php` for an example.

One last comment regarding the questionnaire. The HTML tag <form> was included in the PHP script `quest.head.inc` and the [Submit] and [Reset] buttons of the questionnaire in `quest.bottom.inc`. So, the person coding the questionnaire can concentrate only on the questions and the options for the responses. Changes to the system will not force updates to the questionnaire code.

After questionnaire submission the PHP script `confirmation.php` is called. It receives the answers and updates both MySQL tables (*Flag1*=1 in the respondents table) and the responses text files.

Finally, WSMS1.1 includes a link to phpMyAdmin, but this software is not strictly necessary but certainly is very helpful. You can always manipulate MySQL tables in a command session. In particular phpMyAdmin allows you to export tables in formats directly readable as a spreadsheet.

6 Summary

Web based Survey Mailer System (WSMS) was developed by David J. Solomon, at Michigan State University. It consists of a set of PHP and HTML scripts that allows the administrator to send out personalized emails to potential respondents with links to a questionnaire on the web and capture their answers anonymously. The system allows you to send reminder emails to those who have not filled out the survey. It is based on PHP and MySQL, both open source software and available for Windows and Linux. WSMS1.1 is an extension of the system which reduces the necessary modifications to the system scripts and concentrate the changes in solely HTML documents written by the users. This extension facilitates the exploitation of WSMS by persons less knowledgeable with computer systems.

WSMS1.1 may be thought of as modular with a *Base Page* connecting to the different tasks to carry out the survey. It includes control variables to specify the people from the email list you want to reach. This feature is useful when it is not possible to send out all the emails at once. WSMS1.1 includes a *Consent Page* where subjects are asked for their consent to participate or not in the survey. In addition, the system now captures the answers into MySQL tables as well as text files. The responses are accessible from the *Base Page*, although if the questionnaire is too long, this option may be of no practical use. For this reason, WSMS1.1 also includes a link to the MySQL manager phpMyAdmin. The feature may be ignored or removed if the software is not present in the server.

We think that WSMS1.1 may be extended to help the user in the HTML coding of the questionnaire and generating automatically the list of used variables. Our experience is that most WSMS users are not very familiar with the different languages/software of WSMS. In fact, that was one of the main motivations of extending the original WSMS. The purpose was to make WSMS a usable *tool* for researchers who use web-based surveys as a resource for their research.

Acknowledgement

WSMS1.1 was developed at the Center for Quality and Productivity Improvement, at the University of Wisconsin–Madison, as research tool for the *Information Technology Workforce Project* (ITWF, 2002) supported by the NSF Information Technology Workforce Initiative (NSF EIA-0120092).

Thanks go to Peter Hoonakker who pointed WSMS to me and for all the discussion we had on what kind of needs WSMS1.1 should satisfy. Many thanks go as well to reviewers of the document. Their recommendations made this manual readable.

My sincere appreciation and respect to the Open Source Community.

References

[1] Solomon, David J. (2001). Conducting web-based surveys. *Practical Assessment, Research & Evaluation*, 7(19). (`http://ericae.net/pare/getvn.asp?v=7&n=19`).

[2] Hoonakker, et. al (2003). CQPI Technical Report Series. UW-Madison. Submitted.

[3] *Free Software Foundation* `http://www.gnu.org`

[4] Apache. *The Apache Software Foundation.* `http://www.apache.org`

[5] *PHP*. Another project of The Apache Software Foundation. `http://www.php.net.`

[6] *MySQL*. The Open Source Database. `http://www.mysql.com.`

[7] *Open Source Initiative*. `http://www.opensource.org`

[8] *XAMPP* A very easy to install Apache distribution containing MySQL, PHP and Perl. Just download, extract and start. At the moment there are two XAMPP distributions: Linux and Windows. *...Apache Friends ...* `http://www.apachefriends.org`

[9] *Sokkit*. Sokkit (and PHPTriad in earlier versions) provide a Windows-specific platform for building and serving web sites and applications written in PHP. `http://www.sokkit.net`.

[10] *phpMyAdmin*. Intended to handle the adminstration of MySQL over the web. `http://www.phpwizard.net/projects/phpMyAdmin/`

[11] *K-Meleon* is a lite Web browser. Fast, with minimal interface and it is fully standards-compliant. `http://kmeleon.sourceforge.net`.

Appendix

A.1 WSMS1.1 files

In this section we present a list of all the WSMS1.1 files. Most of the scripts are commented along the code. The asterisk `*` after the file name indicates that the script needs to be changed to better fit particular projects. Special attention should be paid to those files with `html` extension. These are the pages displayed to the survey respondents.

`already.html*` Displays the message that the questionnaire has already been answered. Change the message most appropriate to your project.

`answers.php` Displays the Create Variables Form. Calls `answers.table.php` after submission.

`answers.table.php` From the tab-formatted file `quest.var.dat` creates the responses MySQL table and the text files (`quest.out.dat` and `survey.out.bak`) which host the responses. It also generates the PHP script `quest.link.php` that links the Base Page to `quest.out.dat`.

`base.php*` Display the Base Page with links to all the forms and text representation of the respondents and responses MySQL tables. The page includes a link to phpMyAdmin that you may need to change accordingly or remove it at all.

`confirmation.php` This PHP script maybe the most important script after the questionnaire. It checks whether the person is authorized to answer the questionnaire or not, or if he/she has already done so. If the answers are accepted then the respondents and responses tables are updated as well as the corresponding text files.

`connect.inc` Generated by `connect.settings.php` after submission of the Home Definitions and MySQL Form. Contains the user and password used by MySQL and the home directory and url of the site.

`connect.php` Presents the Home Definitions and MySQL Form. `connect.settings.php` is called after submission.

`connect.settings.php` Captures the user and password used by MySQL and the home directory and url of the site. Saves the information in

`consent.html*` This is the Consent Page. Modify it accordingly your project. Change `consent.php` accordingly if another file name wants to be used.

`consent.bottom.inc` The scrip is appended to the Consent Page. It includes the YES/NO buttons and the link back to Base Page.

`consent.head.inc` The script precedes the Consent Page. Verifies whether the visitor is an actual respondent or not. Checks as well if the subject has already answered the questionnaire or not.

`consent.link.php` Updates the respondents MySQL table. Sends the respondent to the Questionnaire or to the Miss you Form depending on his/her response.

`consent.php` The script simply joins together the header and the bottom scripts to the consent page. See next two files.

`cookiecheck.inc` PHP script that verifies if a visitor is authorized or not checking whether the cookie was set or not logging in. The script is included in most of the WSMS forms.

`copyright.html` HTML format copyright statement.

`copyright.txt` Text format copyright statement.

`format.inc` Set of function and variable definitions for useful when formatting.

`id.inc` Removes the leading digit from persons id and changes it to xxx if the survey is carried out anonymously.

`index.html*` General information page displayed by default if the browser is pointed to the url with no specific page. This files prevent to display a list of files.

`logcheck.php*` Verifies the login password. If this is correct sets a cookie and takes you to the Base Page, otherwise asks you for the password again. You have to change it to the password you want to use.

`login.html` Login page to WSMS, calls for `logcheck.php`.

`missyou.html*` Change it to the message you want displayed when a person does not agree to participate in the survey.

`presentation.email.php` Calls for `presentation.sendit.php`. This script displays the Presentation Email Form.

`presentation.sendit.php` Script that sends out presentation emails to those persons in the respondents table.

`quest.bottom.inc` Code that appends the link back to Base Page when the questionnaire is visited from this page. It is called by `quest.php`.

`quest.head.inc` Code that includes a link back to Base Page when the questionnaire is visited from this page or if the person is authorized to answer the questionnaire.

`quest.html` The questionnaire code. Called by `quest.php`.

`quest.link.php` Generated by `answers.table.php`. Links the Base Page with the questionnaire.

`quest.out.dat` This is the responses text file. Created by the Variables Form and updated whenever a questionnaire is submitted.

`quest.php` This script simply calls `quest.head.inc`, `quest.html` and `quest.bottom.inc`.

`quest.var.dat` Tab delimited text file with the variables list.

`quest.var.php` Generated by `answers.table.php`. It is a PHP script which joins in a single array all the variables. It is called by `confirmation.php`.

`quest.var.txt` Variables list example.

`README.txt` General comments.

`respondents.dat*` Tab delimited text file with sample, sample consecutive number, first name, last name, and email address of the persons included in the survey. This file is used to create the respondents MySQL table.

`respondents.inc` The script reads the respondents MySQL table and saves data in `respondents.out.dat` text file.

`respondents.link.php` Generated by `respondents.table.php`. Links to the respondents text file called from Base Page.

`respondents.out.dat` Created by `respondents.table.php`. This is a text file of the respondents MySQL table.

`respondents.php` Displays the Upload Respondents Page. Calls `respondents.table.php`.

`respondents.table.php` Generates a PHP script with a link to this file from Base Page. The script also creates/updates the respondents MySQL table and its text file version `respondents.out.dat`.

`respondents.txt` An example text file used to illustrate a respondents tab delimited data.

`thanks.html*` Called by `confirmation.php` displays a "Thank you" message after a person has filled out the questionnaire. You may want to change the message.

`tryagain.php` Display a message saying the password provided is wrong and asks again for it.

`url.email.php` Displays the form to send out emails with the url of the questionnaire. After submission calls `ur.sendit.php`.

`url.sendit.php` Sends out the emails with the url of the questionnaire including the person id and whether the person's id is kept or the answers are consider anonymous.

`wrongid.html*` Message shown when the link to the Consent Page or Questionnaire has wrong id.

`wsms.png` A image file with the WSMS "signature", included in the questionnaire (`quest.html`).

A.2 WSMS1.1 download

WSMS1.1 (zipped) scripts can be downloaded from

`http://cqpi2.engr.wisc.edu/wsms/docs/wsms_1.1/code.zip`

A.3 Copyright

If you have questions concerning WSMS or how it is licensed, contact the author at:

David J. Solomon, Ph.D
A-202 E. Fee Hall
Michigan State University
East Lansing, Mi 48824
(517) 353-2037
dsolomon@msu.edu

The same GPL applies to WSMS1.1

For questions regarding WSMS1.1 and how it is licensed, please contact the developer at:

Ernesto Barrios
CQPI / UW–Madison
610 Walnut Street 575
Madison, WI. 53726
ebarrios@engr.wisc.edu