RUNNING HEAD: Website Development

Website Development for Automation of the Leave/Travel System and to Interface a Database

By

Linda M'mayi

Fort Valley State University, GA

August 11, 2005

Supervisor: Mari Herrera

Summer Internship in Science and Technology (SIST) - 2005

Fermi National Accelerator Laboratory

Batavia, IL

Website Development for Automation of the Leave/Travel System and to Interface a Database

Abstract

The Computing Division (CD) at Fermilab receives several leave and travel requests every year from its employees. These requests are handled by the administrative staff who have to follow certain procedures to get approval for the requestors. To help facilitate the process of making leave/travel requests and the work done by the staff, steps are being taken to gradually automate the process. This paper discusses some of the steps taken to achieve this goal including development of a travel website that contains documentation for the travel processing staff, and development of online forms for making requests. The website and the online forms were created using HTML and Macromedia Dreamweaver MX 2004®. This paper also discusses development of websites using software known as CodeCharge Studio 2.3® to interface an underlying database.

Introduction

Fermi National Accelerator Laboratory (Fermilab) is a research laboratory that focuses on advancing the understanding of the fundamental nature of matter and energy by providing leadership and resources for qualified researchers to conduct basic research at the frontiers of high energy physics and related disciplines. In order to accomplish this critical mission, Fermilab relies on the expertise of several scientists, engineers, and other staff. The lab is divided in to several divisions and sections that each contribute significantly to its mission. One such division is the Computing Division (CD).

During my internship at Fermilab, I was privileged to work as an intern in the Computing Division under the Program Support group. The Computing Division's mission is to play a full part in the overall mission of the laboratory and in particular to proudly develop, innovate, and support excellent and forefront computing solutions and services, recognizing the essential role of cooperation and respect in all interactions between ourselves and with the people and organizations that we work with and serve. The Program Support group, as part of the Computing Division, plays an important part in accomplishing the division's mission by providing Administrative, Publishing, Documentation and Infrastructure support in the Computing Division.

The objective of this paper is to describe the project that I worked on during my internship, that is, the process of developing a website for both the leave/travel system and for interfacing an underlying database. The project can be divided in to two parts: the leave/travel request system website and the database website. The purpose of the leave/travel request subproject was to facilitate the processing of leave and travel requests made by the Computing Division employees by making documentation on how to handle

the requests readily available to staff on a travel website, providing online forms for obtaining desired information from users, and simplifying other general tasks. The purpose of the website/database subproject was to create a website that would serve as a user-friendly interface to an underlying database such as a MySQL database, therefore allowing Program Support staff to easily manipulate data in the database via the web as opposed to having to do so directly in the database itself.

I. Automation of the Leave and Travel System

1. Analysis of the Existing System

When I joined the program support group, the leave and travel system was essentially a manual system. The idea of having an automated leave and travel request system had been discussed and few steps had already been taken to start developing the system. For instance, there was some documentation available online for the travel staff and a simple flowchart that demonstrated how the automated system would work had been drawn. Since this leave/travel request system was in actuality two separate systems; I started by working on the leave request system. Since this automated leave request system was in actuality still in the process of being proposed, I was asked to assist in developing an overview of how the automated system would work.

2. The Leave system

The first stage of my project was to develop a detailed flowchart describing how the automated leave request system would work. In order to do so, I studied an existing flow chart and solicited more information on how leave requests are processed.

2.1. Method Used and How the System Works

Then I proceeded to draw a flowchart using Microsoft Word describing the process. The leave request system would work as follows: First, the leave requestor would complete an online leave request form. The information from the form would be saved in a central database and the requestor's supervisor and colleagues of his/her leave request would be notified of the request (In order for this to work, programmers would need to develop appropriate software). If the request is approved or denied by the supervisor, the requestor is notified. Additionally, if the leave is approved then the information is posted on the CD internal website so that colleagues are aware of who is on leave and for how long. The flowchart drawn is provided in appendix A.

Once this flowchart was complete, I submitted it for further consideration by the CD. Hence this system was put on hold while I worked on the travel system.

3. The Travel System:

The automated travel system is being implemented gradually as the different steps are considered and approved by the supervisors. My first task was to develop web pages that contained online documentation related to the travel request process for the Program Support staff. This involved obtaining suitable material for the website content as well as identifying a suitable design for the website. Since there was information available online on a website, I decided to use information from the website, in addition to the information collected from the travel staff, as the content for the new travel website that I was developing.

3.1 Method

To develop the web site, I used Hypertext Markup Language (HTML) and Macromedia Dreamweaver MX® software (referred to simply as Dreamweaver). Dreamweaver is a commercial software that is used for building web sites and applications. It provides a combination of visual layout tools, application development features, and code editing support, enabling developers and designers to create websites and applications more efficiently (Dreamweaver MX 2004). Dreamweaver allows the users to create websites in the following languages HTML (Hypertext Markup Language), PHP (Hypertext Preprocessor), and ColdFusion, as well as to create Active Server Pages (ASP) that are interactive WebPages. Dreamweaver also allows the use of Cascading Style Sheets (CSS) and JavaScript Style Sheets (JSS) which specify the design styles for the website. It also contains detailed tutorials on how to use the software. It's because of these features that we decided it was an appropriate tool for the task at hand.

3.2 Designing the Website

To begin with, I created the two templates for the index page. The templates had similar content but differed in terms of design. The purpose of this was to allow the Program Support staff to evaluate and determine which template they preferred. Once a suitable design was identified based on feedback, I proceeded to develop the other pages in the selected format. The website consisted of three pages: the Index, the Foreign Travel Management System (FTMS) page, and a page for general travel process information, which were all linked to one another and to the Computing Division's internal web site (CD Internal). The website also included sample letters, forms and any other documentation that the Program Support staff required. When the general content

of the web site was in place, I started to add the additional information and graphics. For instance, I incorporated links to other travel-related sites and sample letters in the website. The sample letters were created in MS Word and imported to the travel web site using Dreamweaver.

3.3 Editing and Publishing the Website

As part of the development process, I forwarded the travel website to the Program Support staff for evaluation. This was to enable them to evaluate the content of the site and inform me of any changes or corrections that they felt were necessary before the website was published. Final editing for the travel web pages was done based on feedback from the Program Support staff and the website was submitted for publication on the server. The folder containing the new travel website was saved on the server and the CD internal website was updated such that the new travel website was linked to the CD internal website.

4. Developing Electronic Forms

The next step was to develop online forms that would be used not only for travel related tasks but would also be used as templates for developing forms for other purposes such as submitting requests for using different facilities and registration for conferences. Since the idea of using electronic forms had not officially been approved, my supervisor and I had to write a project plan so that I could get approval to include electronic forms on the website. A project plan gives an overview of what the actually project is, its purpose and advantages, and how it would be implemented. In this case, the project plan, described the purpose of the electronic forms, what forms would be used for and their benefits, how they would be developed and implemented. The project plan was then

submitted for consideration. In the mean time, I proceeded to develop a sample electronic form to be used for placing online requests particularly requests for conference room reservations.

4.1 Method

The electronic form was developed using Dreamweaver software, HTML and Perl. The online form system comprises of four HTML pages (a form page, a return page, a log page and an error log page) and a Perl script.

4.2 Description of How the Electronic Forms work

Each component of the online form serves a different purpose. The form page (example in appendix B) provides the user interface and consists of different fields to be used to collect information from the users. The return page (example in appendix B) is the confirmation page that the user gets after successfully submitting their request. It also provides the user with information such as whom to contact if they need further assistance. The log page (example in appendix C) is a blank web page where information from the form would be placed and that would be updated automatically each time a new request is made. The error log page keeps track of errors that the users encounter while using the request form and is accessible only by the web developer. The Perl script processes information from the submitted form, saves and updates the log page, and returns the return page whenever a new request was submitted. It also simultaneously sends emails to the Program Support staff notifying them of the new request.

The online requests system would works as follows: when a user wants to place a request they would access the CD Internal page where there would be a link to the electronic request form. The user would have to complete the form and click the submit

button to submit the request. Once the user successfully submits a request, he/she would get a response (return page) indicating that his/her request has been received. At the same time, an email is sent to the program support staff informing them that a request has been made. The Perl code used to generate this online request system is available in appendix E.

4.3 Testing and Debugging the Perl script

Several modifications had to be done before the script was able to work as expected. The server URLs had to be accurate as well as the name of the form objects such as textfields and text areas. When all the variables had been checked, I ran a sample test of the script with some test data and confirmed that it was working as expected. The script was designed in such a way that future use of the script would require very little modification of the script therefore it will be very useful for creating other forms in future.

4.4 Documentation

After creating the sample form, I worked on writing documentation. Writing precise documentation is an important aspect of any project. It provides the opportunity to explain clearly what the developed program does and how to use it, and also serves as a valuable resource for future reference. In my case, the documentation elaborated the significance of having the forms online, how the forms worked and how I went about developing the forms. This documentation will enable the Program Support staff to use the developed sample form as a template for other forms they may require in future.

5. Discussion

With the travel website in place, the travel staff can easily access information they need via the web as opposed to having to manually locate files and folders when they need some clarifications. The travel website will be used by Programs Support staff for references and also to facilitate training of new employees who join the travel staff. Because the proposal to include the electronic forms in the travel website was denied; the form was not included in the travel website. Nevertheless, we were able to use the form for other purposes such as registration of participants for a conference to be held at Fermilab later this year.

II. Website Development Using Code Charge Studio 2.3®

1. Overview

The Program Support group was interested in finding software that could be used to develop websites for manipulating data in a database. One piece of software that was being considered was CodeCharge Studio 2.3®. The next phase of my project was to develop web interfaces for manipulating the data in a database using the specified software so that the group could see whether or not it was suitable for achieving the goal.

1.1 Method

I used a piece of software known as CodeCharge Studio 2.3® which I will refer to as CodeCharge. CodeCharge is commercially available software that is used as an effective tool for creating database-driven Web applications with minimal amount of coding. It includes an Application Builder that enables the user to convert his/her database (MS Access, MS SQL, MySQL, Oracle, etc.) into a working Web application

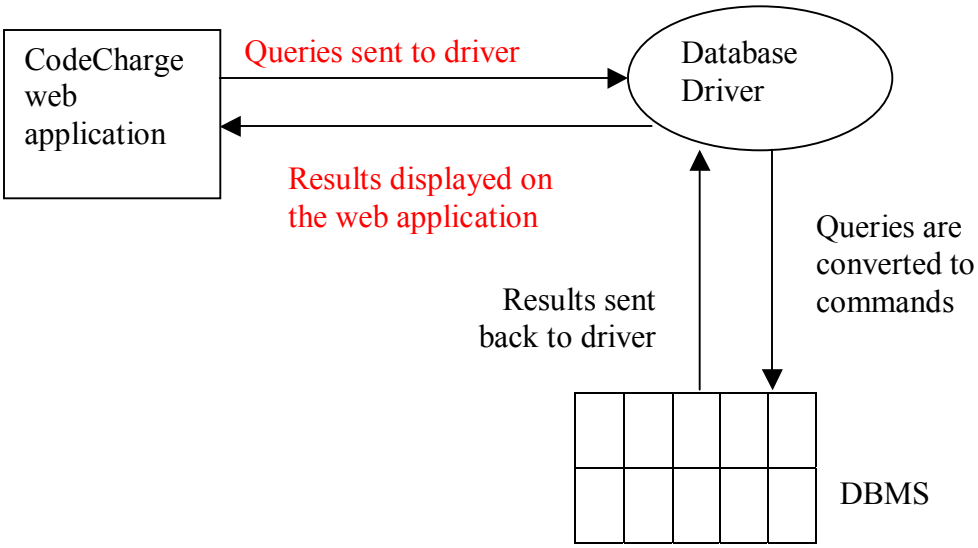
with login-protected user access to database search, list, view and update functions; a Query Builder for visually specifying database tables, fields and their relations; and a Microsoft FrontPage Add-in for converting Microsoft FrontPage in to a web application development environment.

The websites created by CodeCharge are in the following languages: C#, PHP, VBScript, Perl and JSP. The advantages of using CodeCharge to develop web interfaces include (1) reducing the need for hand coding since the software does most of the actual coding; (2) software is accompanied by a detailed tutorial that guides the user on how to use it; and (3) CodeCharge, as opposed to using the regular database forms and pages such as those in MS Access, can interface a variety of databases whereas the database forms and pages are restricted to the particular database type. CodeCharge works in conjunction with the appropriate database ODBC driver to establish a connection between the website and the database.

1.2 How Code Charge Works

Open Database Connectivity (ODBC) is a standard database access method developed by the SQL (Structured Query Language) Access group in 1992. The goal of ODBC is to make it possible to access any data from any application, regardless of which database management system (DBMS) is handling the data. ODBC manages this by inserting a middle layer, called a database driver, between an application and the DBMS. The purpose of this layer is to translate the application's data queries into commands that the DBMS understands. For this to work, both the application and the DBMS must be *ODBC-compliant,* that is, the application must be capable of issuing ODBC commands and the DBMS must be capable of responding to them (Webopedia). CodeCharge Studio

uses this concept to connect web pages to the corresponding database. The figure below outlines how ODBC works.



Once the CodeCharge Studio software has been installed on the machine, simply open the software to start using it. To create a new web interface, the user can easily select the appropriate options for creating a new project, and follow the steps provided by the application builder. The application builder guides one through a step by step process to develop the website allowing one to select which functions (such as Delete, Add, Edit) from a given set of functions to incorporate in his/her web site. The user also selects which data should be displayed by specifying the tables and fields from which data should be retrieved. In addition, the application builder helps the user configure the connection to be used to access the database. This requires the user to enter the correct server path and URL for the server hosting the database, and also select the appropriate ODBC driver for the connection based on the type of underlying database. Once the user clicks the Finish button of the application builder, they are presented with a template of what the web page looks like. Any further editing can be done at this design stage. To

publish the page, the user can select the "Live Page" option on the menu and the page will be published on the specified server.

1.3 Developing a website Using CodeCharge Studio 2.3 ®

For my assignment, I first had to create a sample Microsoft Access database and a sample website to familiarize myself with the CodeCharge software. I created the database with two tables – StudentInfo and AcademicInfo - then entered sample data in to the tables. I used CodeCharge to create a web interface for the database.

1.4 Results

Using the application builder, I followed the steps and was able to successfully create a website and edit it in the Design mode. An example of a website created using CodeCharge is provided in appendix D**.**

1.5 Discussion

Although CodeCharge was quite effective for developing the websites desired, I wasn't able to proceed with the actual implementation of my project (development of web interfaces for the MySQL database) due to complications involving the set up of the existing MySQL database. For me to able to proceed with my assignment, configuration changes would have to be made on the database. At the time of writing this paper, configurations were yet to be changed hence I was unable to complete this project.

Nevertheless, CodeCharge proved to be a very useful piece of software. When using it the user didn't have to do much coding since CodeCharge did most of the work. It's easy to use and is accompanied by very detailed tutorials as well as readily available online support. However, the CodeCharge software is very specific in terms of requirements needed to set up the environment for developing, testing and implementing

website therefore making it somewhat complicated to use. There are several requirements that have to be met to have the suitable environment and these requirements are different for each language and database.

Overall Conclusion

One of the tasks handled by the Program Support group in the Computing Division is the processing of leave and travel requests for the division. Since the Computing Division consists of over 200 employees, the Program Support groups receives numerous leave and travel requests every year. It's for this reason that the idea of automating the system was put forward. Automating the leave and travel system is a complex and gradual process. My project this summer helped facilitate the automation process by: (1) providing a website containing leave and travel documentation for Program Support staff , (2) creating electronic forms that can be used for a wide range of tasks, and (3) since I was able to use CodeCharge Studio to develop web applications for interfacing databases, in future if a database is created for storing leave and travel information or for any other purpose, CodeCharge can also be used to create web interfaces for the databases. Although the automation of the leave and travel system is still far from complete, once its fully developed and implemented, it will benefit not just the Computing Division but Fermilab as a whole because the developed system can serve as a template for the other divisions.

References

Castro, E. (2001). *Perl and CGI for the World Wide Web*. CA: Peachpit Press.

CodeCharge Studio 2.3, Retrieved 07/26/05, from http://www.yessoftware.com/products/product_detail.php?product_id=1

Lemay, L. (1995). *Teach Yourself Web Publishing With HTML in a Week*. IN: Sams Publishing.

Macromedia Dreamweaver MX 2004 at a Glance, Retrieved 07/18/05, from http://www.macromedia.com/software/dreamweaver/productinfo/overview/

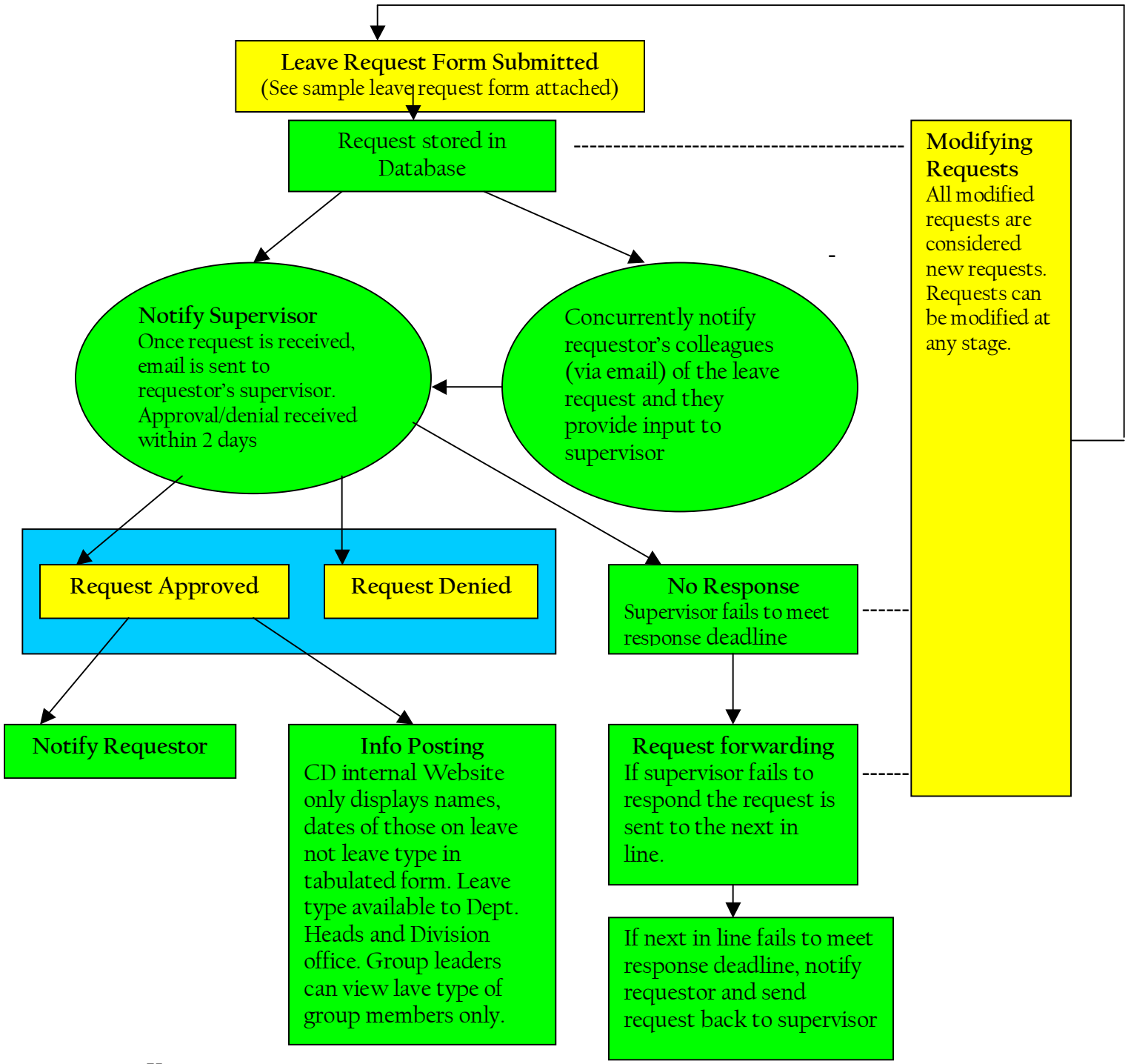
Open Database Connectivity, Retrieved 07/18/05, from *Webopedia* http://www.webopedia.com/TERM/O/ODBC.html

Acknowledgements

I wish to extend my uttermost appreciation to God and to the following organizations and individuals for enabling me to participate in this internship program and for making my experience at Fermilab both intellectually and socially stimulating.

- Fermilab
- Computing Division and in particular the Program Support group
- Mrs. Dianne Engram, Dr. Elliott McCrory, and SIST Committee
- Supervisor: Mari Herrera
- Mentors: Cosmore Sylvester, Jamieson Olsen, Dr. Davenport and Dr. Herman White
- Fellow Interns

Appendix A: Flowchart

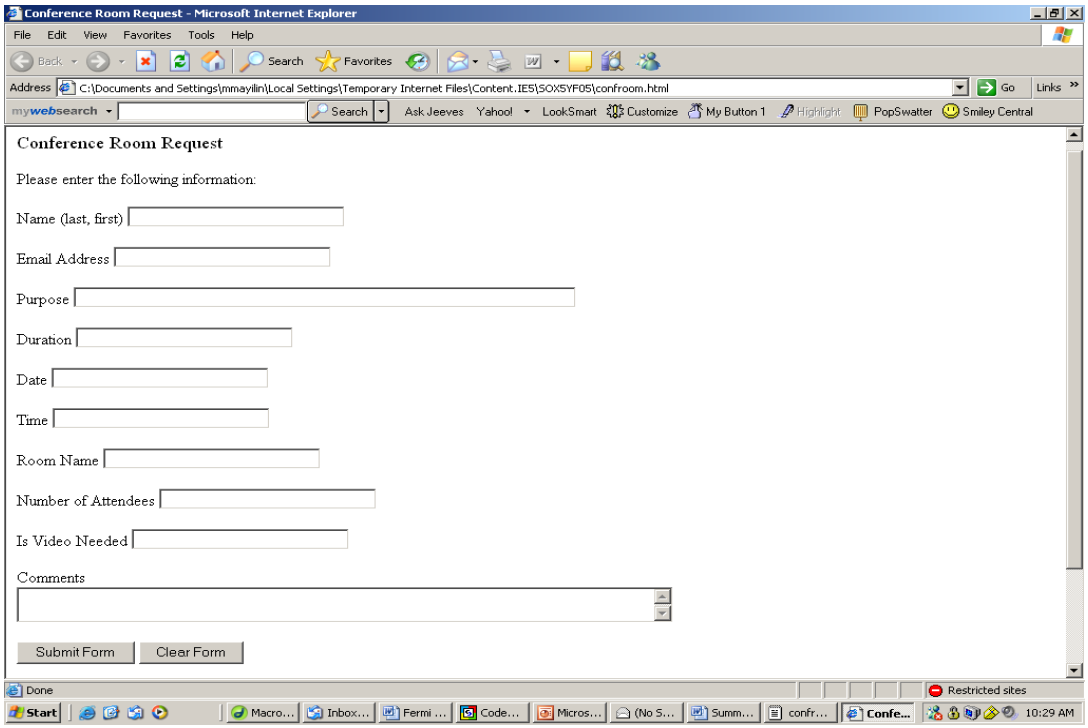


Key

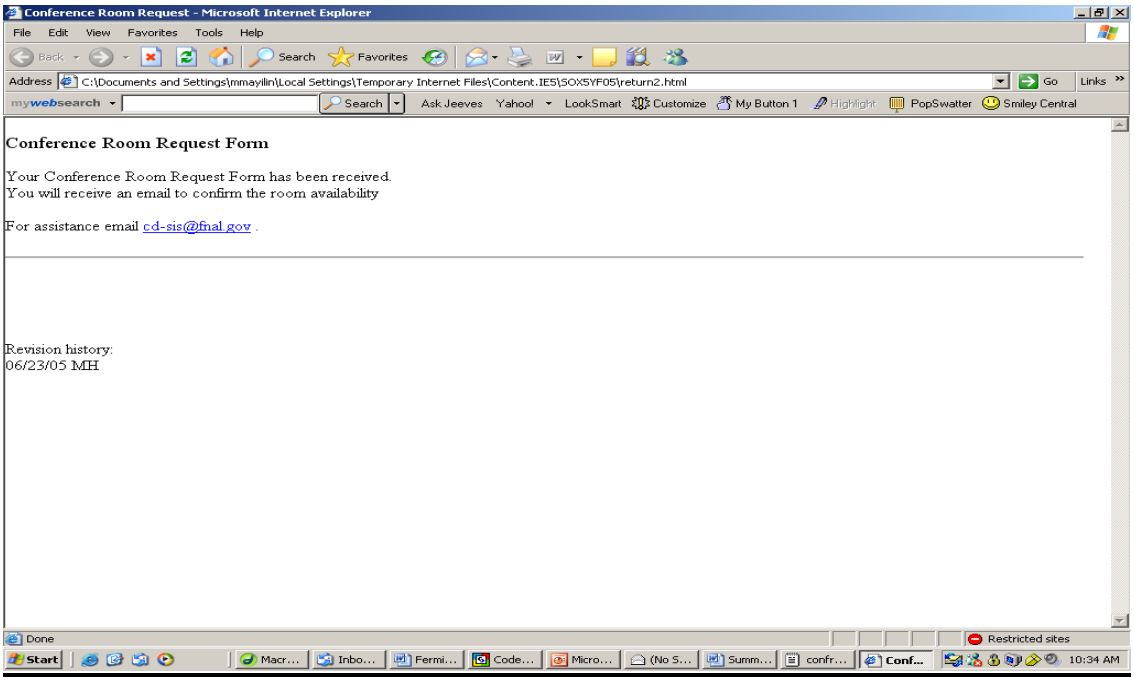
- Step requires electronic signature
- Step requires a user interface
- Workflow software actions, email notifications, and other underlying processes

Appendix B:

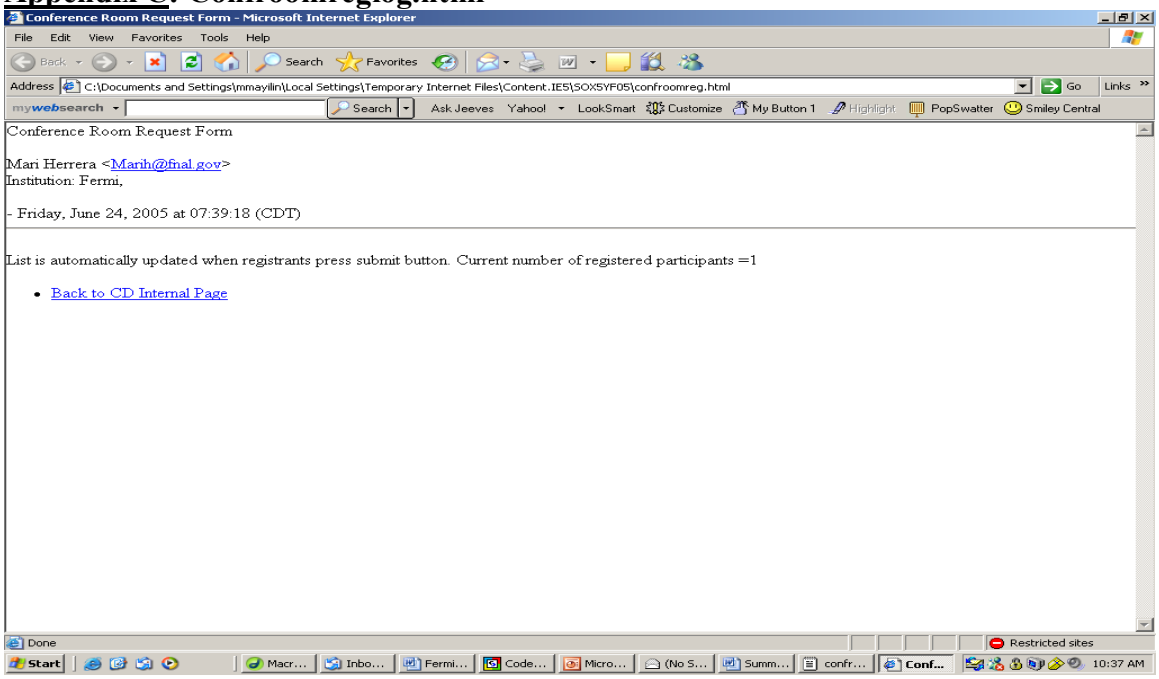
Confroom.html



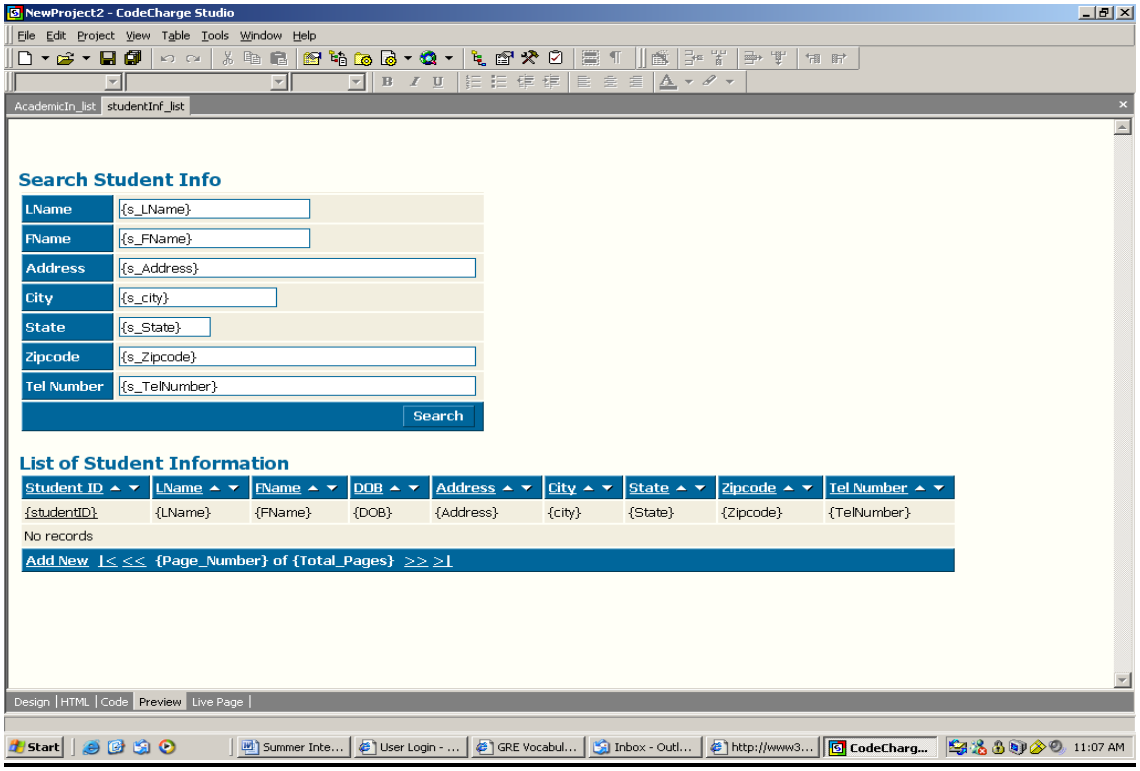
Return.html



Appendix C: Confroomreglog.html



Appendix D: Sample Website Created Using CodeCharge Studio



Appendix E: Confroom.pl

```
#! /usr/local/bin/perl
BEGIN {
require "/afs/fnal/ups/etc/setups.pl";
ups::use_prod("perl_dbd_mysql","v2_9003", "SETUP_PERL_DBD_MYSQL");
$^W=0;
}
use CGI qw/:standard/;

# Set Variables
$guestbookurl = "http://home.fnal.gov/~marih/public_html/forms_requests/confroomreg.html";
$guestbookreal = "/afs/fnal.gov/files/home/room3/marih/public_html/forms_requests/confroomreg.html";
$guestlog =  "/afs/fnal.gov/files/home/room3/marih/public_html/forms_requests/confroomreglog.html";
$returnpageurl = "http://cdinternal.fnal.gov/DOE_Workshop/doe8345/return2.html";
$cgiurl = "http://computing.fnal.gov/cgi-bin/aheavey/confroom.pl";
$date_command = "/usr/bin/date";

# Set Your Options:
$mail = 1;              # 1 = Yes; 0 = No  #changed to 1
$uselog = 1;            # 1 = Yes; 0 = No
$linkmail = 1;          # 1 = Yes; 0 = No
$separator = 1;         # 1 = <hr>; 0 = <p>
$redirection = 1;       # 1 = Yes; 0 = No
$entry_order = 1;       # 1 = Newest entries added first;
                        # 0 = Newest Entries added last.
$remote_mail = 1;       # 1 = Yes; 0 = No
$allow_html = 1;        # 1 = Yes; 0 = No
$line_breaks = 0; # 1 = Yes; 0 = No

# If you answered 1 to $mail or $remote_mail you will need to fill out
# these variables below:
$mailprog = '/usr/lib/sendmail';  #left as is; same as in bfnalformmail anyway 8/3/04
$recipient = 'marih@fnal.gov, mmayilin@fnal.gov';
#########################################################
#first open for reading, read in whole thing, increment the value, then replace whole thing

open (GUEST,"$guestbookreal") || die "Can't Open $guestbookreal: $!\n";
flock (GUEST, 2)  || Error ('lock', 'file');
@guestlines = <GUEST>;
foreach my $startline (@guestlines)
{
 $firstword=substr($startline, 0, 7);
 if ($firstword eq 'Current')
 {
  #print "original line: ", $startline, "\n";
  @splitline = split(/=/ , $startline);
  $splitline[1]++;
  $startline = $splitline[0]."=".$splitline[1]."\n";
  #print "updated line is ", $startline, "\n";
 }
}
#print "closing file for reading now", "\n";
close GUEST;
#####################################################################
```

```
open (GUEST,">$guestbookreal") || die "Can't Open $guestbookreal: $!\n";
flock (GUEST, 2)  || Error ('lock', 'file');
print GUEST "\n";
print GUEST @guestlines;
close GUEST;
##########################################################################

# Get the Date for Entry
$date = `$date_command +"%A, %B %d, %Y at %T (%Z)"`; chop($date);
$shortdate = `$date_command +"%D %T %Z"`; chop($shortdate);

# Get the input
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});

# Split the name-value pairs
@pairs = split(/&/, $buffer);

foreach $pair (@pairs) {
  ($name, $value) = split(/=/, $pair);

  # Un-Webify plus signs and %-encoding
  $value =~ tr/+/ /;
  $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
  $value =~ s/<!--(.|\n)*-->//g;

  if ($allow_html != 1) {
    $value =~ s/<([^>]|\n)*>//g;
  }

  $FORM{$name} = $value;
}
# Print the Blank Response Subroutines
#commented out &no_comments unless $FORM{'comments'};
#&no_name unless $FORM{'realname'};
##########################################################################

# Begin the Editing of the Guestbook File
open (FILE,"$guestbookreal") || die "Can't Open $guestbookreal: $!\n";
@LINES=<FILE>;
close(FILE);
$SIZE=@LINES;

# Open Link File to Output
open (GUEST,">$guestbookreal") || die "Can't Open $guestbookreal: $!\n";
for ($i=0;$i<=$SIZE;$i++) {
  $_=$LINES[$i];
  if (/<!--begin-->/) {
    if ($entry_order eq '1') {
      print GUEST "<!--begin-->\n";
    }
    if ($line_breaks == 1) {
      $FORM{'comments'} =~ s/\cM\n/<br>\n/g;
    }
    print GUEST "<b>$FORM{'comments'}</b><br>\n";
```

```
    if ($FORM{'url'}) {
      print GUEST "<a href=\"$FORM{'url'}\">$FORM{'realname'}</a>"; }
    else {
      print GUEST "$FORM{'realname'}"; }
# In following, changed username to email 4x
    if ( $FORM{'email'} ){
      if ($linkmail eq '1') {
        print GUEST " \&lt;<a href=\"mailto:$FORM{'email'}\">";
        print GUEST "$FORM{'email'}</a>\&gt;";
      }
      else {
        print GUEST " &lt;$FORM{'email'}&gt;";
      }
    }
    print GUEST "<br>\n";

    if ( $FORM{'Purpose'} ){
      print GUEST "Purpose: $FORM{'Purpose'},";
    }
    print GUEST "<br>\n";

    if ( $FORM{'Room'} ){
      print GUEST "Room Name: $FORM{'Room'}";
    }
    print GUEST "<br>\n";

    if ( $FORM{'Attend'} ){
      print GUEST "Number of Attendees: $FORM{'Attend'}";
    }
     print GUEST "<br>\n";

     if ( $FORM{'Video_pref'} ){
        print GUEST "Is Video Needed: $FORM{'Video_pref'}";
     }
      print GUEST "<br>\n";

     if ( $FORM{'Comments'} ){
        print GUEST "Comments: $FORM{'Comments'}";
     }
      print GUEST "<br>\n";

     if ( $FORM{'parallel2'} ){
        print GUEST "Parallel session (2nd): $FORM{'parallel2'}";
     }
      print GUEST "<br>\n";


    if ($separator eq '1') {
      print GUEST " - $date<hr>\n\n";
    }
    else {
      print GUEST " - $date<p>\n\n";
    }
    print GUEST "<br>\n";

    if ($entry_order eq '0') {
```

```
        print GUEST "<!--begin-->\n";
      }
    }
    else {
      print GUEST $_;
    }
}
close (GUEST);
# Log The Entry
if ($uselog eq '1') {
  &log('entry');
}
############################################################
# Mail Option
if ($mail eq '1') {
  open (MAIL, "|$mailprog $recipient") || die "Can't open $mailprog!\n";
  print MAIL "Reply-to: $FORM{'email'} ($FORM{'realname'})\n";
  print MAIL "From: $FORM{'email'} ($FORM{'realname'})\n";
  print MAIL "Subject: Conference Room Request\n\n";
  print MAIL "There is a new Conference Room Request.\n\n";
  print MAIL "-----------------------------------------------------\n";
  #print MAIL "$FORM{'comments'}\n";
  print MAIL "$FORM{'realname'}";

  if ( $FORM{'email'} ){
    print MAIL " <$FORM{'email'}>";
  }
  print MAIL "\n";
    if ( $FORM{'Purpose'} ){
      print MAIL "Purpose: $FORM{'Purpose'},";
   }
   print MAIL "\n";
    if ( $FORM{'Duration'} ){
      print MAIL "Duration: $FORM{'Duration'},";
    }
   print MAIL "\n";
     if ( $FORM{'Time'} ){
      print MAIL "Start Time: $FORM{'Time'},";
    }
   print MAIL "\n";
    if ( $FORM{'Date'} ){
      print MAIL "Date: $FORM{'Date'},";
    }
   print MAIL "\n";
    if ( $FORM{'Room'} ){
      print MAIL "Room Name: $FORM{'Room'}";
    }
   print MAIL "\n";
    if ( $FORM{'Comments'} ){
      print MAIL "Comments: $FORM{'Comments'}";
    }
   print MAIL "\n";
      if ( $FORM{'Video_pref'} ){
       print MAIL "Is Video Needed: $FORM{'Video_pref'}";
     }
   print MAIL "\n";
```

```
    if ( $FORM{'Attend'} ){
      print MAIL "Number of Attendees (1st): $FORM{'Attend'}";
    }
  print MAIL "\n";
  print MAIL " - $date\n";
  print MAIL "-------------------------------------------------------\n";
  close (MAIL);
}
#########################################################################
if ($remote_mail eq '1' && $FORM{'email'}) {
  open (MAIL, "|$mailprog -t") || die "Can't open $mailprog!\n";
  print MAIL "To: $FORM{'email'}\n";
  print MAIL "From: $recipient\n";
  print MAIL "Subject: Conference Room Request\n\n";
  print MAIL "You have successfully requested a Conference Room\n";
  print MAIL "You will receive an email to confirm the room availability\n\n";
  print MAIL "With question contact cd-sis@fnal.gov\n\n";
  print MAIL "-------------------------------------------------------\n";
   #print MAIL "$FORM{'comments'}\n";
   print MAIL "$FORM{'realname'}";

   if ( $FORM{'email'} ){
     print MAIL " <$FORM{'email'}>";
   }
  print MAIL "\n";
  print MAIL " - $date\n";
  print MAIL "-------------------------------------------------------\n";

  close (MAIL);
}
#############################################################
# Print Out Initial Output Location Heading
# problem with redirection, try commenting out.
if ($redirection eq '1') {
  print "Location: $returnpageurl\n\n";
}
else {
  &no_redirection;
}#######################
# Subroutines
sub no_comments {
  print "Content-type: text/html\n\n";
  print "<html><head><title>No Comments</title></head>\n";
  print "<body><h1>Your Comments appear to be blank</h1>\n";
  print "The comment section in the guestbook fillout form appears\n";
  print "to be blank and therefore the Guestbook Addition was not\n";
  print "added.  Please enter your comments below.<p>\n";
  print "<form method=POST action=\"$cgiurl\">\n";
  print "Your Name:<input type=text name=\"realname\" size=30 ";
  print "value=\"$FORM{'realname'}\"><br>\n";
  print "E-Mail: <input type=text name=\"username\"";
  print "value=\"$FORM{'username'}\" size=40><br>\n";
  print "City: <input type=text name=\"city\" value=\"$FORM{'city'}\" ";
  print "size=15>, State: <input type=text name=\"state\" ";
  print "value=\"$FORM{'state'}\" size=15> Country: <input type=text ";
```

```
  print "name=\"country\" value=\"$FORM{'country'}\" size=15><p>\n";
  print "Comments:<br>\n";
  print "<textarea name=\"comments\" COLS=60 ROWS=4></textarea><p>\n";
  print "<input type=submit> * <input type=reset></form><hr>\n";
  print "Return to the <a href=\"$returnpageurl\">Return Page</a>.";
  print "\n</body></html>\n";

  # Log The Error
  if ($uselog eq '1') {
    &log('no_comments');
  }

  exit;}
sub no_name {
  print "Content-type: text/html\n\n";
  print "<html><head><title>No Name</title></head>\n";
  print "<body><h1>Your Name appears to be blank</h1>\n";
  print "The Name Section in the guestbook fill out form appears to\n";
  print "be blank and therefore your entry to the guestbook was not\n";
  print "added.  Please add your name in the blank below.<p>\n";
  print "<form method=POST action=\"$cgiurl\">\n";
  print "Your Name:<input type=text name=\"realname\" size=30><br>\n";
  print "E-Mail: <input type=text name=\"username\"";
  print " value=\"$FORM{'username'}\" size=40><br>\n";
  print "City: <input type=text name=\"city\" value=\"$FORM{'city'}\" ";
  print "size=15>, State: <input type=text name=\"state\" ";
  print "value=\"$FORM{'state'}\" size=2> Country: <input type=text ";
  print "value=USA name=\"country\" value=\"$FORM{'country'}\" ";
  print "size=15><p>\n";
  print "Comments have been retained.<p>\n";
  print "<input type=hidden name=\"comments\" ";
  print "value=\"$FORM{'comments'}\">\n";
  print "<input type=submit> * <input type=reset><hr>\n";
  print "Return to the <a href=\"$returnpageurl\">Return Page</a>.";
  print "\n</body></html>\n";
  # Log The Error
  if ($uselog eq '1') {
    &log('no_name');
  }
  exit;
}
# Log the Entry or Error
sub log {
  $log_type = $_[0];
  open (LOG, ">>$guestlog");
  if ($log_type eq 'entry') {
    print LOG "$ENV{'REMOTE_HOST'} - [$shortdate]<br>\n";
  }
  elsif ($log_type eq 'no_name') {
    print LOG "$ENV{'REMOTE_HOST'} - [$shortdate] - ERR: No Name<br>\n";
  }
  elsif ($log_type eq 'no_comments') {
    print LOG "$ENV{'REMOTE_HOST'} - [$shortdate] - ERR: No ";
    print LOG "Comments<br>\n";
  }}
# Redirection Option
```

```
sub no_redirection {

  # Print Beginning of HTML
  print "Content-Type: text/html\n\n";
  print "<html><head><title>Thank You</title></head>\n";
  print "<body><h1>Thank You for registering for the OSG Operations Meeting</h1>\n";

  # Print Response
  print "Thank you for registering.  Your entry has\n";
  print "been added to the meeting participants page.<hr>\n";
  print "Here is what you submitted:<p>\n";
  print "<b>$FORM{'comments'}</b><br>\n";

  if ($FORM{'url'}) {
    print "<a href=\"$FORM{'url'}\">$FORM{'realname'}</a>";
  }
  else {
    print "$FORM{'realname'}";
  }
  if ( $FORM{'email'} ){
    if ($linkmail eq '1') {
      print " &lt;<a href=\"mailto:$FORM{'email'}\">";
      print "$FORM{'email'}</a>&gt;";
    }
    else {
      print " &lt;$FORM{'email'}&gt;";
    }
  }
  print "<br>\n";

    print " $FORM{'country'}";
  if ( $FORM{'country'} ){
  }
  print " - $date<p>\n";
  # Print End of HTML
  print "<hr>\n";
  print "<a href=\"$returnpageurl\">Back to the Return page.</a>\n";        print "- You may need to reload
it when you get there to see your\n";
  print "entry.\n";
  print "</body></html>\n";

  exit;
}
```