2000

The Incident Dispatching and Tracking System

Thomas J. Braden
University of North Florida

THE INCIDENT DISPATCHING AND TRACKING SYSTEM

by

Thomas J. Braden

A project submitted to the Department of Computer and Information Sciences in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

April, 2000

The "Incident Dispatching and Tracking System," submitted by Thomas J. Braden in partial fulfillment of the requirements of the degree of Master of Science in Computer and Information Sciences, has been

Approved by:

Signature Deleted

4/21/2000

Project Director: Dr. Ralph Butler Date

Signature Deleted

4/24/2000

Graduate Director: Dr. Robert Roggio Date

Accepted for the Department of Computing Sciences and Engineering:

Signature Deleted

4/21/00

Department Chair: Dr. Judith Solano Date

ACKNOWLEDGEMENT

To my wife, Shanon, and my daughter, Heather, for their loving support.

To my father, Thomas G. Braden, whose pride in what I was trying to accomplish kept me going and never allowed me to quit.

A special thanks to Dr. Ralph Butler, for his assistance and tutelage and to all of my family and friends who helped me test this application.

CONTENTS

ABSTRACT

The Incident Dispatching and Tracking System (IDTS) was developed to facilitate the technical support of an organization. IDTS provides a means by which a user may submit their issue/problem to the system and receive a unique trouble ticket number. This number can then be used to view the status of that record, minimizing the need for inquiries to the technical support team. IDTS also provides a dispatching function that assigns each trouble ticket to a technician that specializes in that type of issue. Finally, there is a reporting function that allows the technical support team to view the current technical status of the organization, generate statistics on previously submitted trouble tickets, and determine areas that may require assistance in the future.

DEVELOPMENT

The purpose of IDTS was to develop a browser based tool that allowed an organization of users to submit problems and issues that would in turn be dispatched to a technician. The application required further functionality, such as the creation of User, Technician, and Administrator lists as well as an audit trail for

each issue being worked on. This functionality was accomplished by creating a series of CGI scripts to display the data that was being manipulated by PERL code (using CGI.pm) and being stored in DBM files. [See Appendix A]

CGI scripting using CGI.pm works well for this type of application because of its ability to collect data and use it to dynamically create HTML forms and tables. Each script starts by "initializing" the page, this includes printing the HTML header, reading the name-value pairs from the calling page, and tying the data in dbm files to hashes. These hashes are then used to fill in fields in the form or to dynamically create popup menus.

After the page has been initialized, an html form is displayed. When the user records or modifies information in the form and clicks the submit button, the data is passed to the next page as name-value pairs and using the POST method.

DBM files were used to store the data because of the easy-to-use PERL interface. By using the dbmopen command, a PERL hash is tied to the data in the DBM file.

Changes to the DBM files are made by manipulating the hash and then closing the DBM file with the dbmclose command. The only issue with using DBM was that the data had to be stored as name-value pairs, but this was easily accomplished by combining all necessary data into one string. [See Appendix B]

Future enhancements to IDTS include an ability to dispatch trouble tickets via e-mail and the ability to generate reports on the performance of the technical support team.

REQUIREMENTS

IDTS requires a web server capable of executing CGI scripts, PERL ver 5.004 or later, and DBM.

INSTALATION

1. Log into the web server as an administrator to the cgi-bin directory.
2. Create a directory under cgi-bin and execute:
   chmod 755 <new directory name>
3. Change Directory to the newly created directory.
4. Copy the contents of the IDTS directory from the CD to the current directory.

5. Change the permissions of all cgi scripts as follows:

   chmod 755 *.cgi

6. Log in to the system using the Administrator ID root with the password root.

7. Use the administrator utilities to add new administrators and change root's password. [See Administrator Guide]

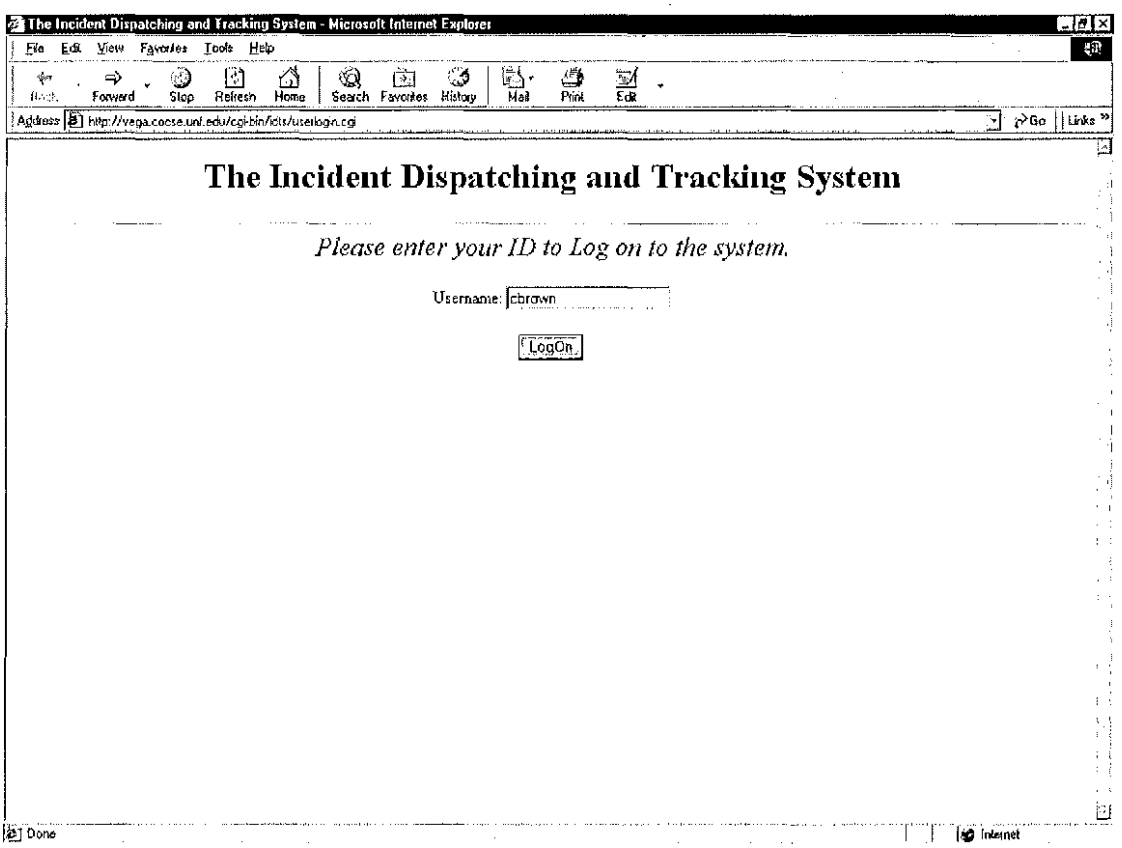
8. If access to the system is ever lost, copy ADMINS.* from the IDTS directory on the CD to the IDTS directory on the web server and use the default root ID (this will overwrite all previously entered Administrator and Technician data).

USER GUIDE

This user guide provides instructions on how to log in to the system, submit a trouble ticket, look-up trouble ticket numbers, and check the status of a submitted ticket.

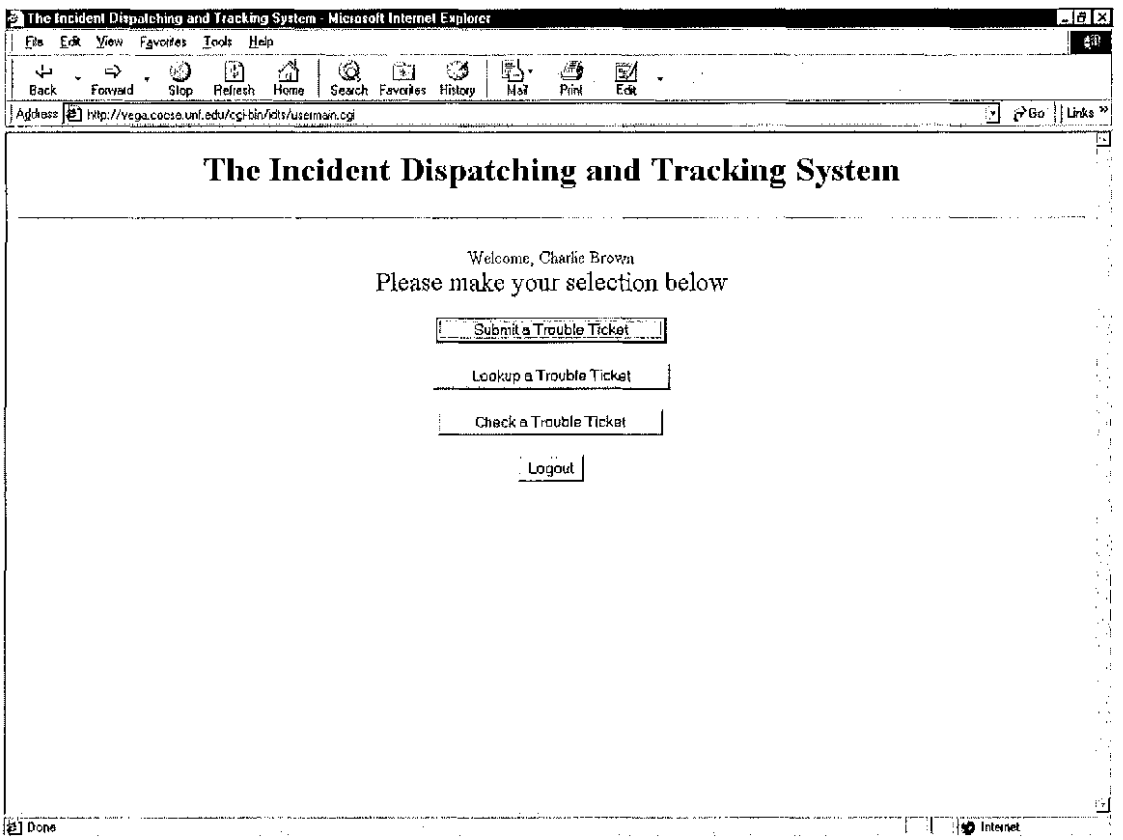
Logging In

To log in, enter your assigned username in the Username box and click the LogOn button.



Main Menu

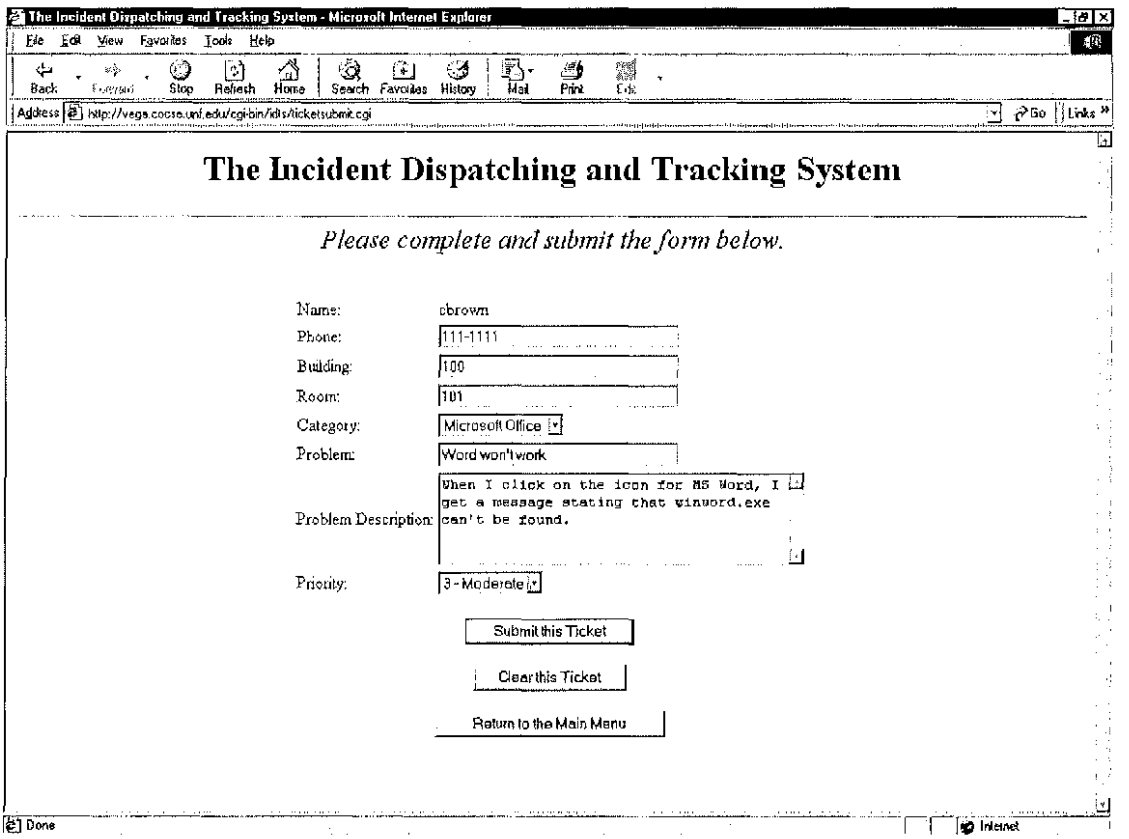
Once logged in, you will be passed to the main menu where you will be greeted with a welcome message. Use this message to verify that you are logged in correctly and with the correct username. This menu contains all of the main functions available to the user, as well as the option to log out of the system.



If you have incorrectly logged in or receive a message stating that the username does not exist, click your browser's back button and try again. Further complications should be communicated to your technical support team.

Submitting a Trouble Ticket

When the Trouble Ticket submission form is displayed, it will contain the user's username as well as the default phone number, building, and room number. These defaults can be changed if warranted by the nature of the issue to be submitted.



The form's fields should be handled as follows:

- Phone, Building, Room - Displays the user's default information but can be changed if warranted.
- Category - Select the category that best suits the nature of the issue being reported.
- Problem - Enter a brief description of the issue.

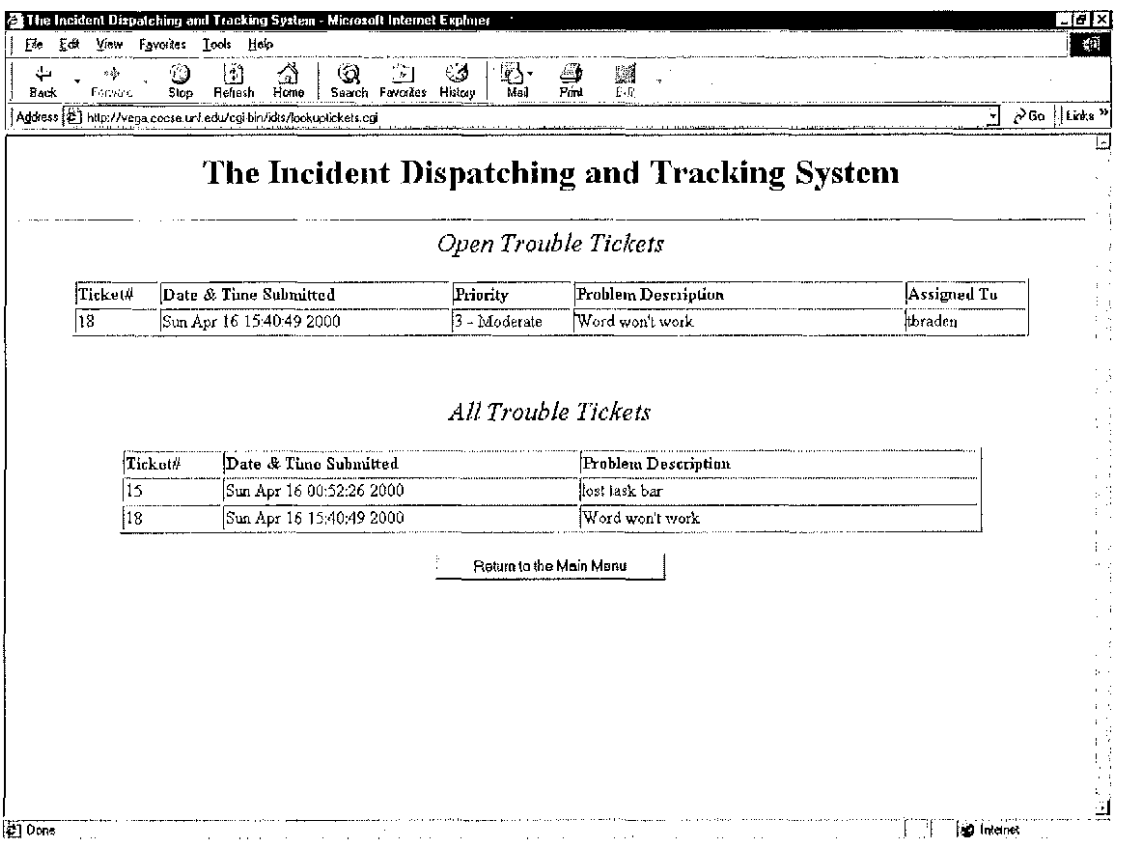
- Problem Description - Enter information that will assist the technician diagnose your problem. This should include a description of how the error occurred and any error messages.
- Priority - Select the priority that best reflects the severity of the issue.

The buttons have the following effects:

- Submit this Ticket - Submits the ticket to be dispatched to a technician. A confirmation will be displayed containing the time the ticket was submitted and a unique trouble ticket number.
- Clear this Ticket - Clears the form and returns it to its default state.

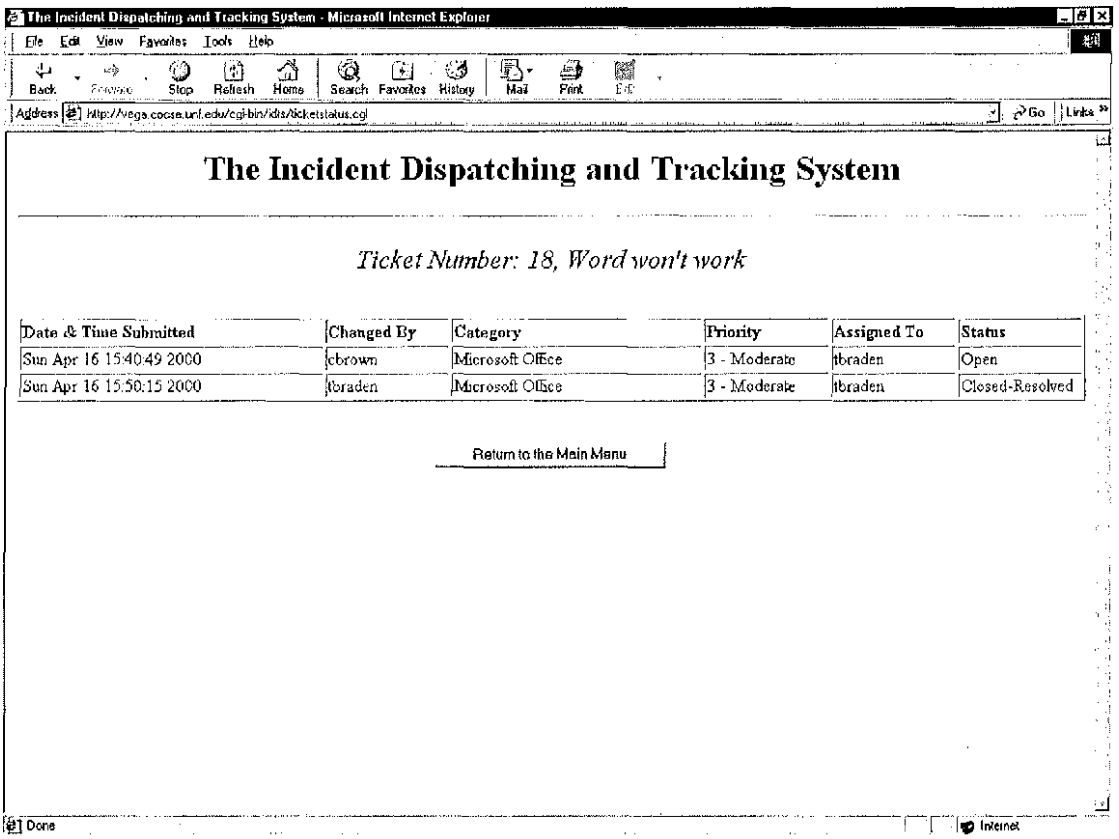
Lookup a Trouble Ticket

This will display a list of Trouble Tickets submitted by the user. The open tickets are displayed first to give a quick status for all pending issues, followed by all tickets ever submitted by this user. These ticket numbers can be used with the Check a Trouble Ticket function to obtain further details of the status of a ticket.



Check a Trouble Ticket

Enter the Trouble Ticket number of the issue you wish to review. A history of changes will be displayed showing who worked the issue and when as well as any changes to the priority and category.

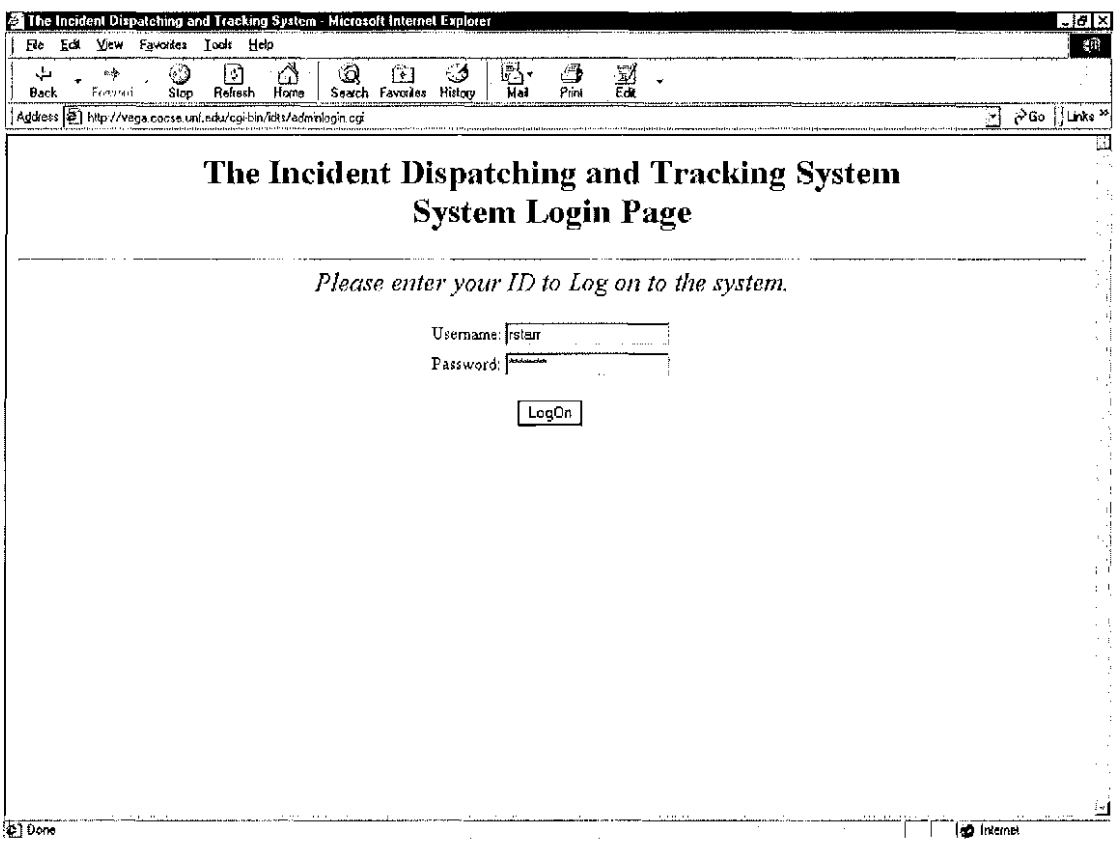


TECHNICIAN GUIDE

This technician guide provides instructions on how to log in to the system, look-up trouble ticket numbers, check the status of a submitted ticket, modify an existing trouble ticket, run a report, and change the password.

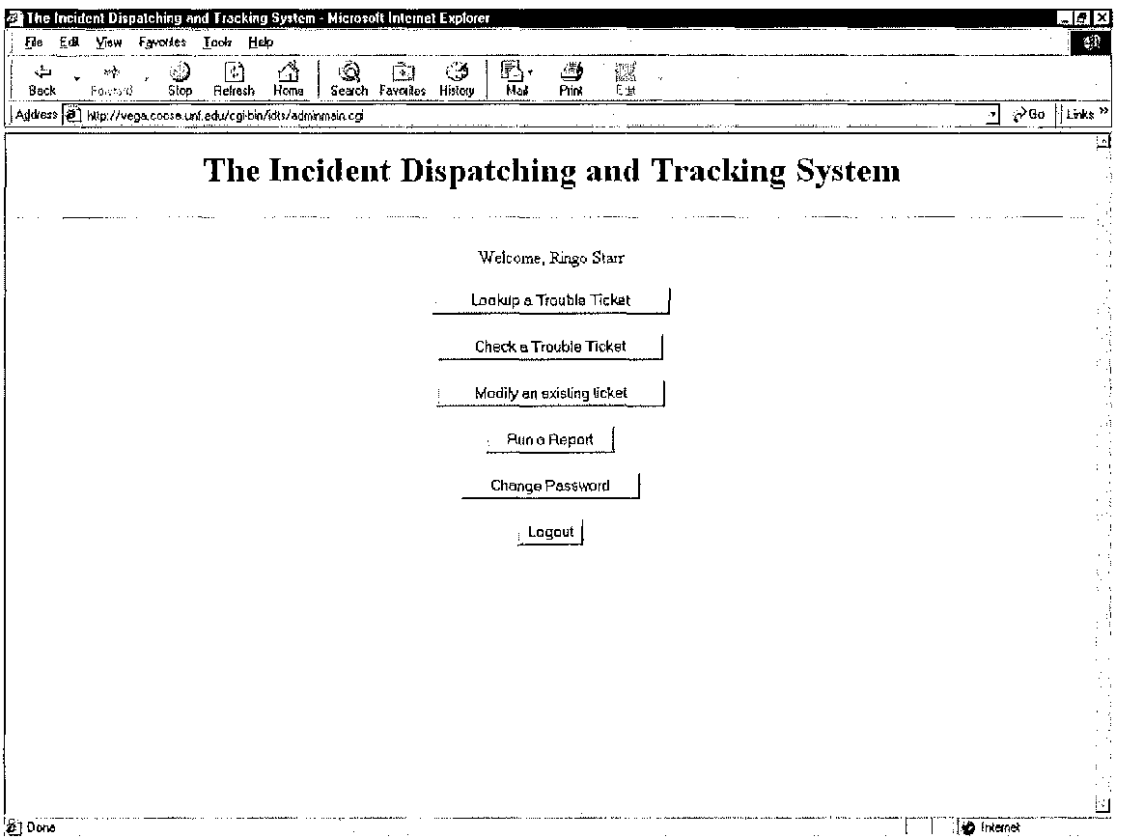
Logging In

To log in, enter your assigned username and password, and click the LogOn button.



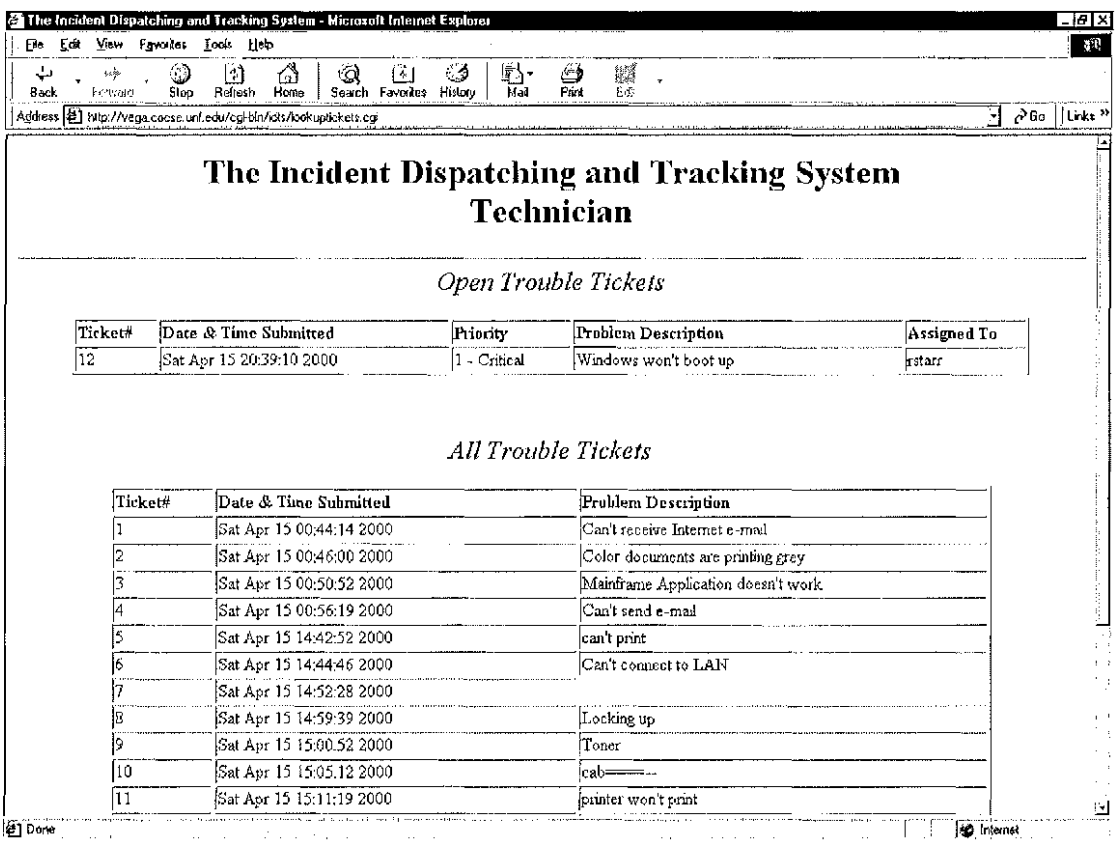
Main Menu

Once logged in, you will be passed to the main menu where you will be greeted with a welcome message. Use this message to verify that you are logged in correctly and with the correct username. This menu contains all of the main functions available to the technician, as well as the option to log out of the system.



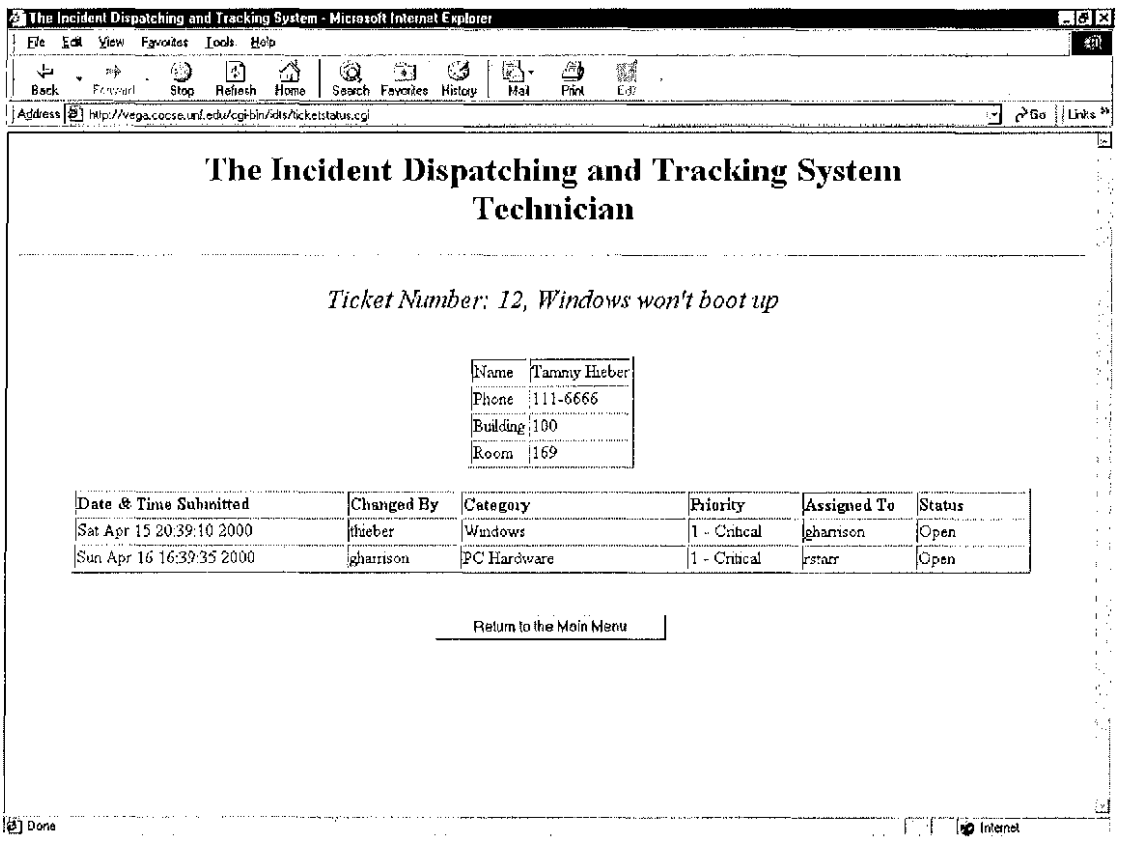
Lookup a Trouble Ticket

This will display a list of Trouble Tickets assigned to the technician. The open tickets are displayed first to give a quick status for all pending issues, followed by a list of all tickets. These ticket numbers can be used with the Check a Trouble Ticket function to obtain further details of the status of a ticket and/or with the Modify a Trouble Ticket to update the status.



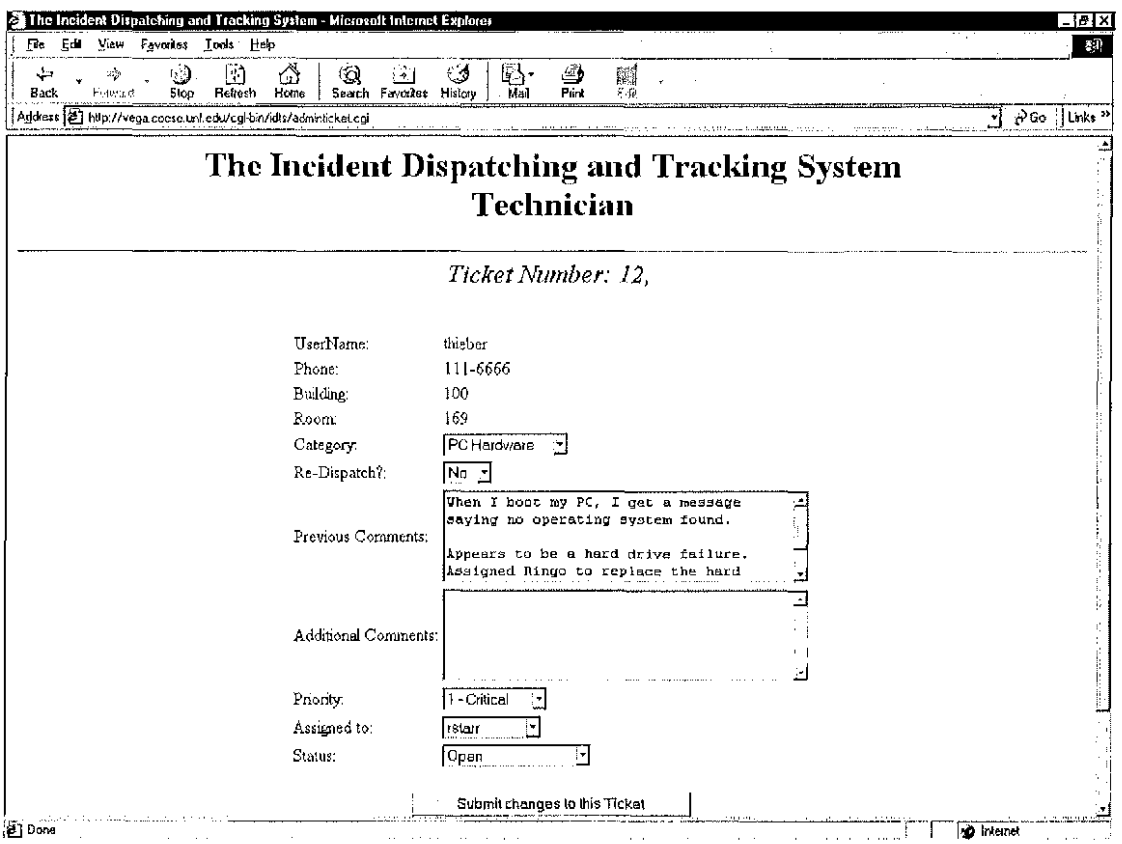
Check a Trouble Ticket

Enter the Trouble Ticket number of the issue you wish to review. A history of changes will be displayed showing who worked the issue and when as well as any changes to the priority and category.



Modify an existing Ticket

When the Modify Trouble Ticket form is displayed, it will contain the username, phone number, building, and room number for the user that submitted the ticket. The form will also display all current information on the ticket, along with all of the previous comments entered on this ticket.



The form's editable fields should be handled as follows:

- Category - Select the category that best suits the nature of the issue being reported.

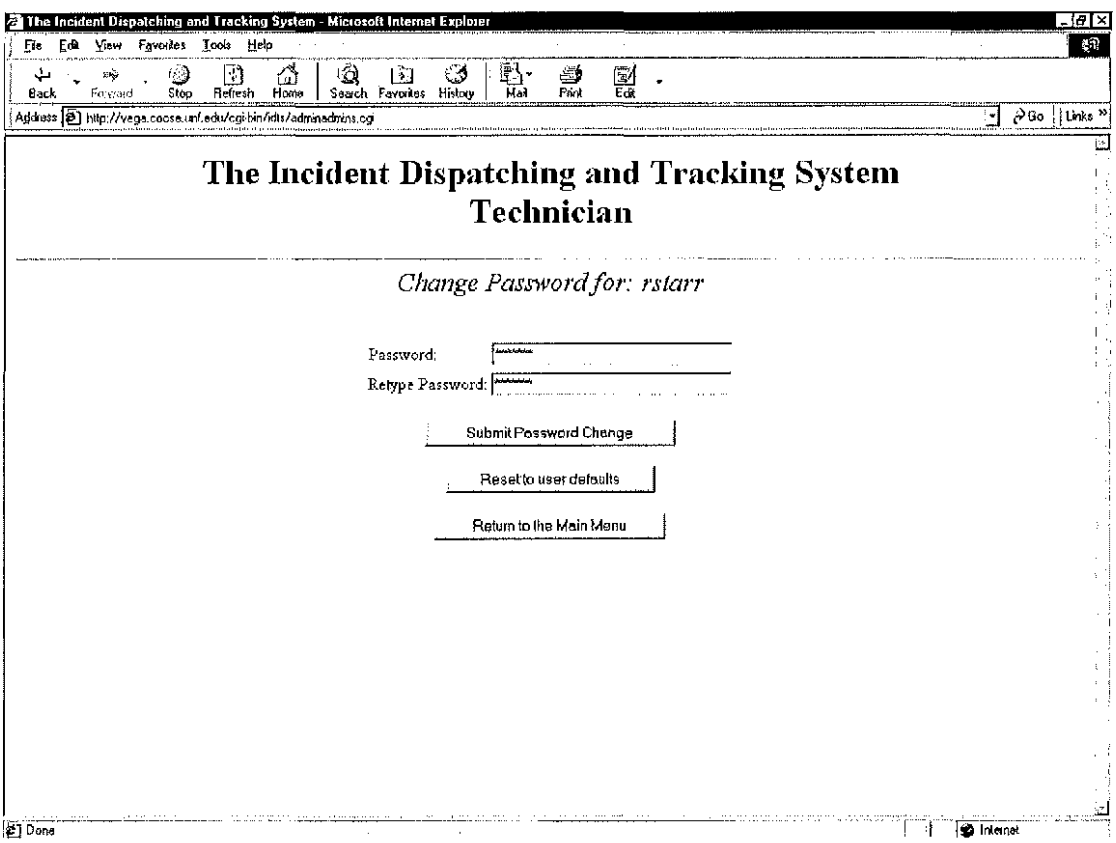
- Re-Dispatch? - If the category is changed, this field can be set to 'Yes' to re-dispatch the issue to a technician that specializes in the new category.
- Previous Comments - This area contains all previous comments entered on this issue. This is not an editable field and changes will not be saved.
- Additional Comments - This area is used to add additional comments to the ticket. These comments will be saved and later viewable in the previous comments field.
- Priority - Select the priority that best reflects the severity of the issue.
- Assigned to - This field can be used to manually change the technician assigned to the issue.
- Status - This field reflects the status of the Trouble Ticket. Options include Open (ticket still being worked on), Suspended (ticket requires further work but cannot be finished at this point), Closed-Resolved (issue has been successfully resolved), Closed-Unsolvable (issue cannot be resolved), and Closed-Irrelevant (inappropriate issue submitted).

Run a Report

Please see the Report Guide.

Change Password

Use this option to change the password. This form requires the technician to enter a new password and then confirm the password by re-typing it.

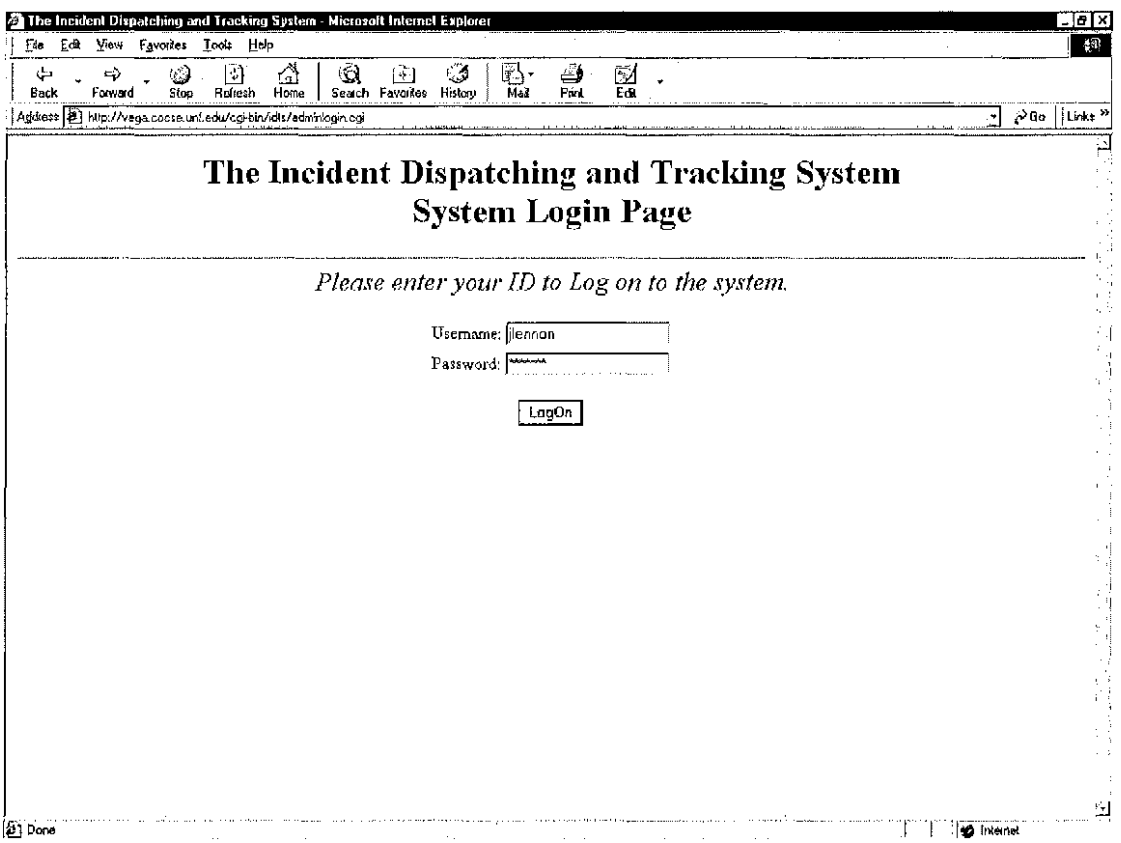


ADMINISTRATOR GUIDE

This administrator guide provides instructions on how to log in to the system, look-up trouble ticket numbers, check the status of a submitted ticket, modify an existing trouble ticket, run a report, and change the password.

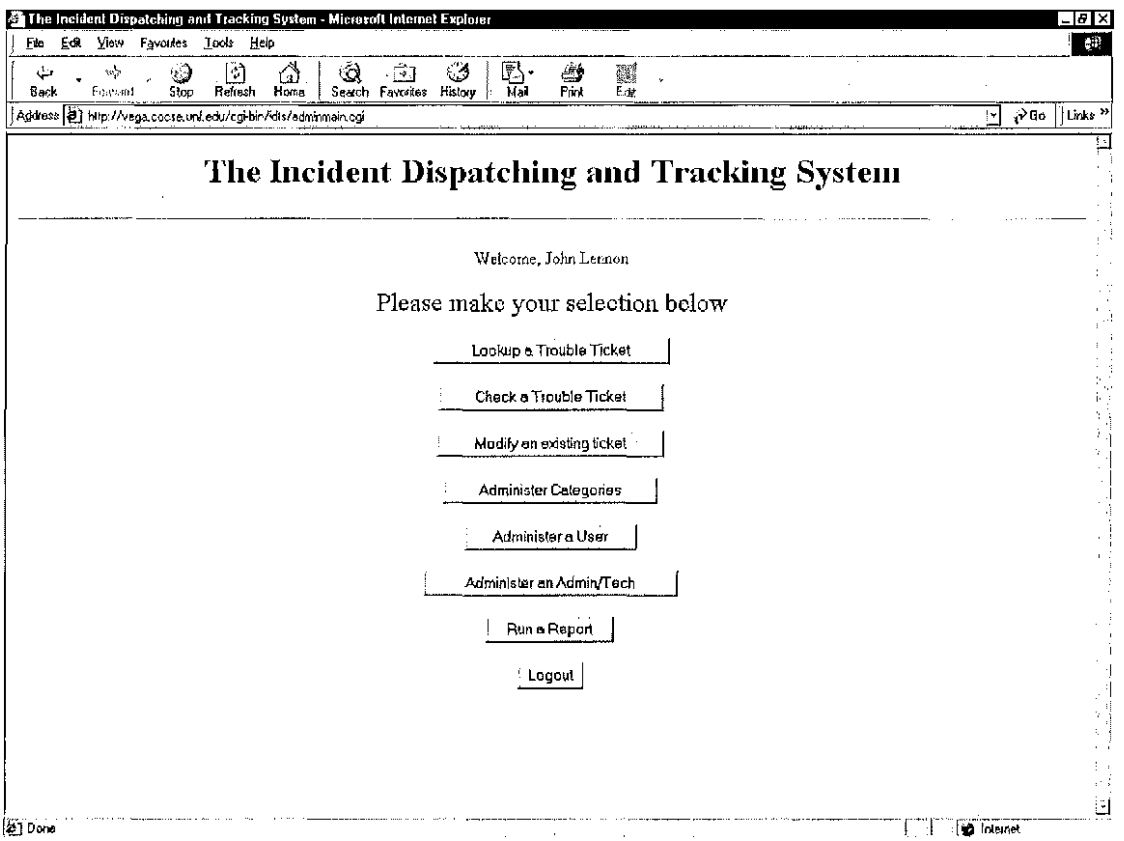
Logging In

To log in, enter your assigned username and password, and click the LogOn button.



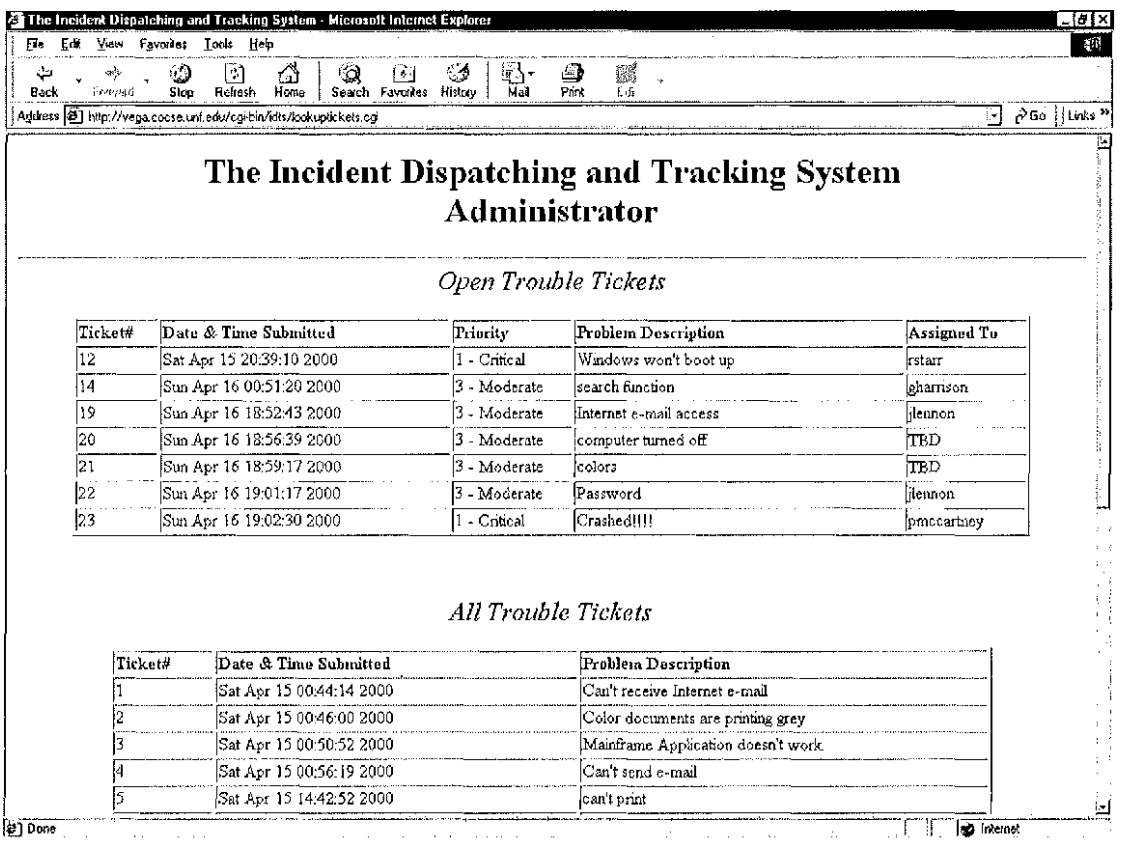
Main Menu

Once logged in, you will be passed to the main menu where you will be greeted with a welcome message. Use this message to verify that you are logged in correctly and with the correct username. This menu contains all of the main functions available to the administrator, as well as the option to log out of the system.



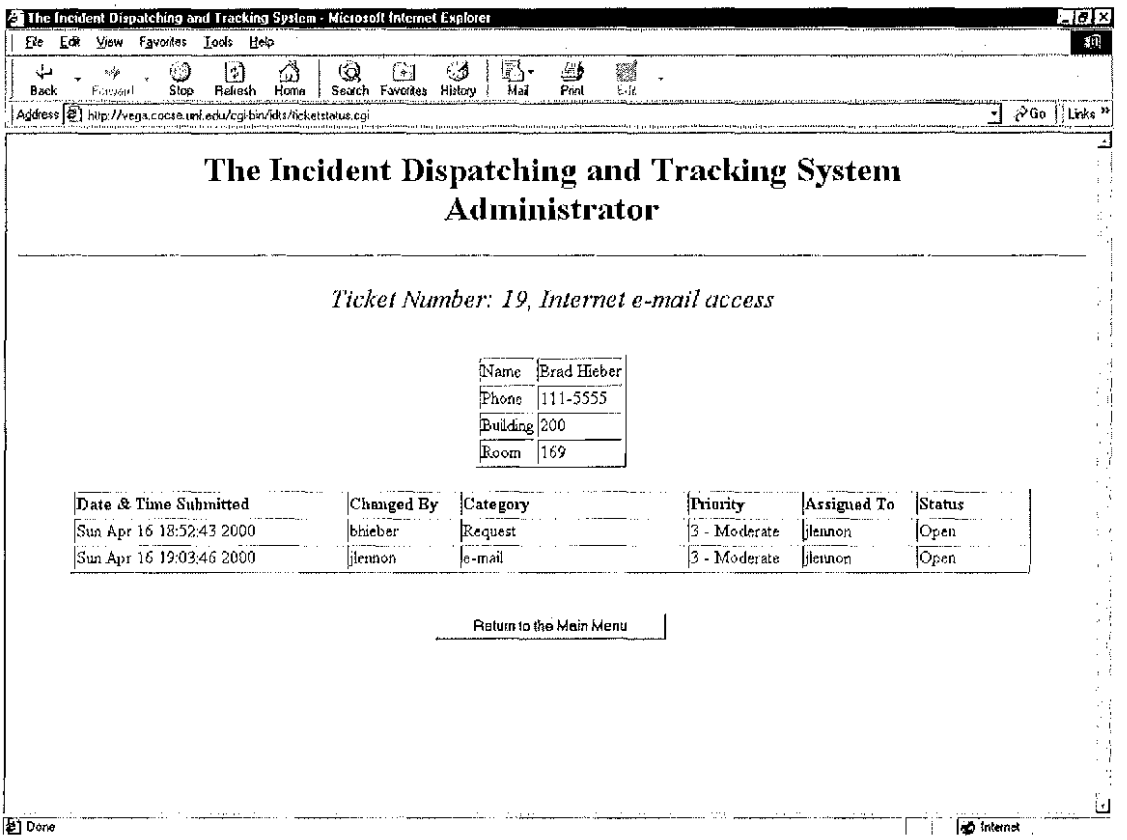
Lookup a Trouble Ticket

This will display a list of all submitted Trouble Tickets. The open tickets are displayed first to give a quick status for all pending issues, followed by a list of all tickets. These ticket numbers can be used with the Check a Trouble Ticket function to obtain further details of the status of a ticket and/or with the Modify a Trouble Ticket to update the status.



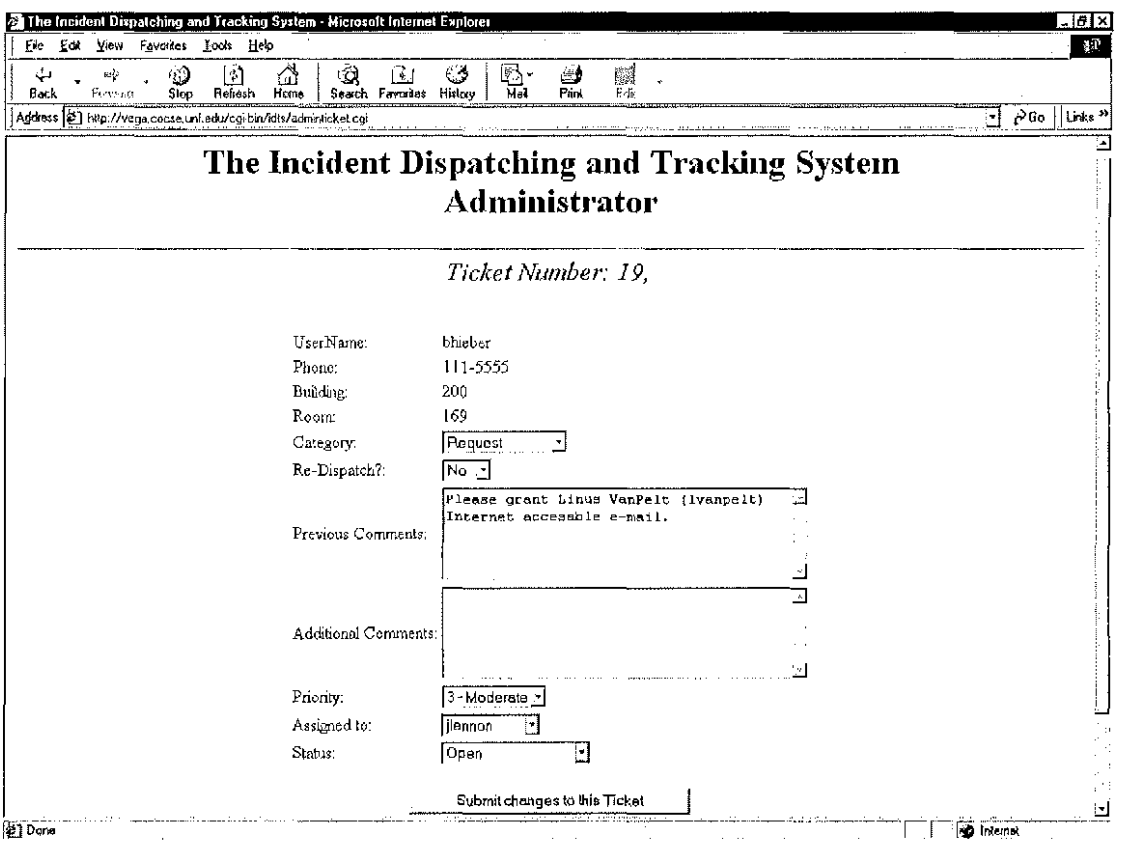
Check a Trouble Ticket

Enter the Trouble Ticket number of the issue you wish to review. A history of changes will be displayed showing who worked the issue and when as well as any changes to the priority and category.



Modify an existing Ticket

When the Modify Trouble Ticket form is displayed, it will contain the username, phone number, building, and room number for the user that submitted the ticket. The form will also display all current information on the ticket, along with all of the previous comments entered on this ticket.



The form's editable fields should be handled as follows:

- Category - Select the category that best suits the nature of the issue being reported.

- Re-Dispatch? - If the category is changed, this field can be set to 'Yes' to re-dispatch the issue to a technician that specializes in the new category.
- Previous Comments - This area contains all previous comments entered on this issue. This is not an editable field and changes will not be saved.
- Additional Comments - This area is used to add additional comments to the ticket. These comments will be saved and later viewable in the previous comments field.
- Priority - Select the priority that best reflects the severity of the issue.
- Assigned to - This field can be used to manually change the technician assigned to the issue.
- Status - This field reflects the status of the Trouble Ticket. Options include Open (ticket still being worked on), Suspended (ticket requires further work but cannot be finished at this point), Closed-Resolved (issue has been successfully resolved), Closed-Unsolvable (issue cannot be resolved), and Closed-Irrelevant (inappropriate issue submitted).

Run a Report

Please see the Report Guide.

Administer Categories

This function is used to add or delete categories used when submitting a trouble ticket or defining a technician's specialties. Use this form as follows:

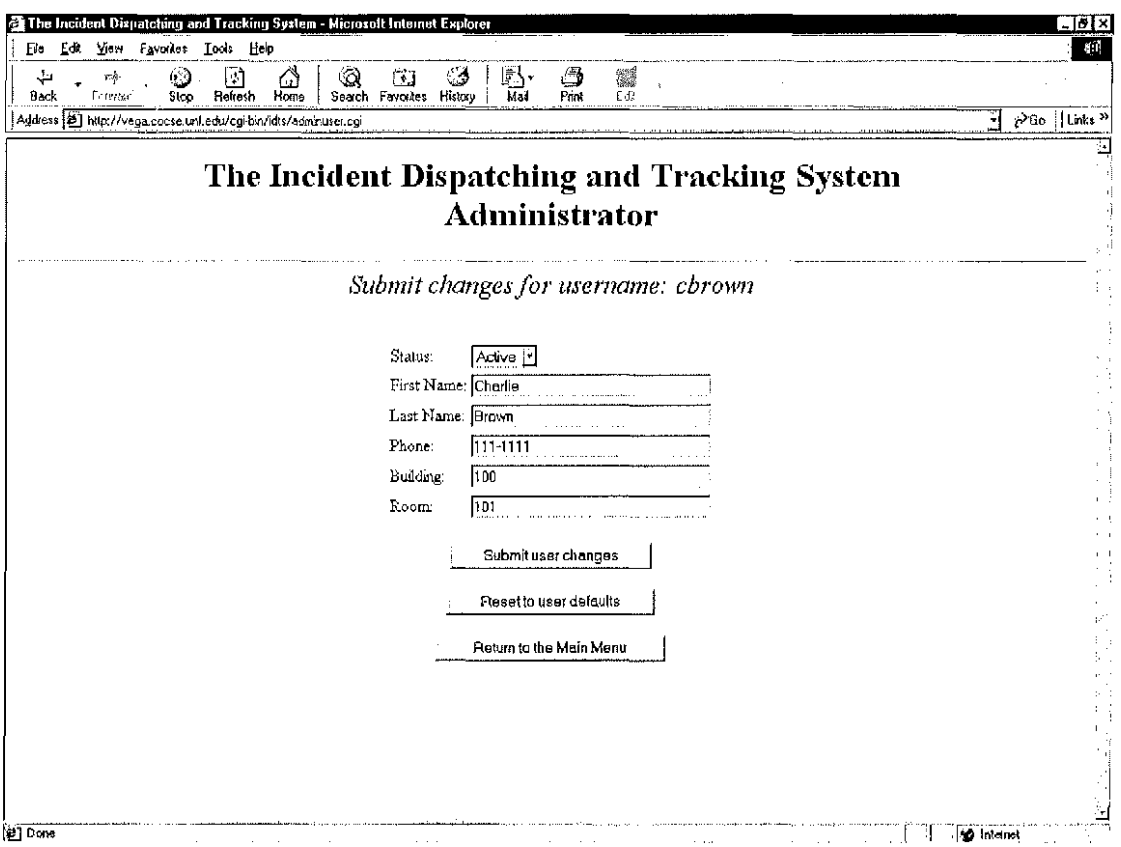
- Add a New Category - Select NEW, then click the Access Category Information button. On the following page, you will be asked for the new category name, enter it and click the Submit New Category button.
- Delete a Category - Select the category from the popup menu, then click the Access Category Information button. On the following page, change the category status to Delete and click the Submit Category Changes button.

Administer a User

This function is used to add or delete Users to the system. Use this form as follows:

- Add a New User - Select NEW, then click the Access User Information button.
- Update an Existing User - Select the Username from the popup menu, then click the Access User Information button.

The user information form will be displayed (blank for new users and containing the current data on existing users).



Use this form as follows:

- Status - Shown only when updating an existing user. Leave as Active or change to Delete to remove a user.
- First & Last Name - The User's name.
- Phone - The user's phone number.
- Building & Room - The user's location.

The buttons have the following effects:

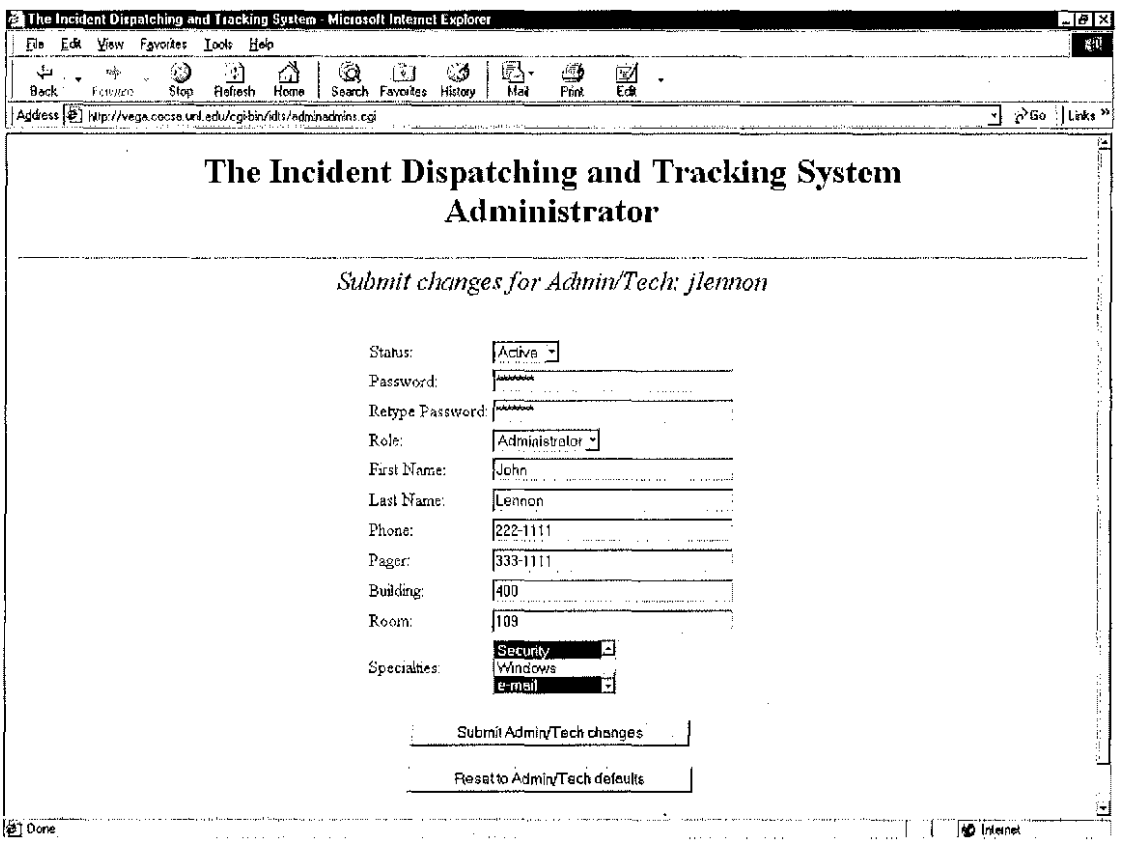
- Submit user changes - Submits the user changes or creates a new user. A confirmation will be displayed containing the time the user update was made.
- Reset to user defaults - Clears the form and returns it to its default state.

Administer an Admin/Tech

This function is used to add or delete Administrators and Technicians to the system. Use this form as follows:

- Add a New Admin/Tech - Select NEW, then click the Access Admin/Tech Information button.
- Update an Existing Admin/Tech - Select the Username from the popup menu, then click the Access Admin/Tech Information button.

The Admin/Tech information form will be displayed (blank for new Admin/Techs and containing the current data on existing Admin/Techs).



Use this form as follows:

- Status - Shown only when updating an existing Admin/Tech. Leave as Active or change to Delete to remove an Admin/Tech.
- Password - Enter the Admin/Tech's password.
- Retype Password - Confirm the password.
- Role - Assign the role of either Administrator or Technician.
- First & Last Name - The Admin/Tech's name.
- Phone - The Admin/Tech's phone number.
- Pager - The Admin/Tech's pager number.
- Building & Room - The Admin/Tech's location.
- Specialties - Assign the types of issues that the Admin/Tech will be working on by selecting one or more of the categories in this list.

The buttons have the following effects:

- Submit Admin/Tech Changes - Submits the Admin/Tech changes or creates a new Admin/Tech. A confirmation will be displayed containing the time the Admin/Tech update was made.
- Reset to Admin/Tech defaults - Clears the form and returns it to its default state.

Report Guide

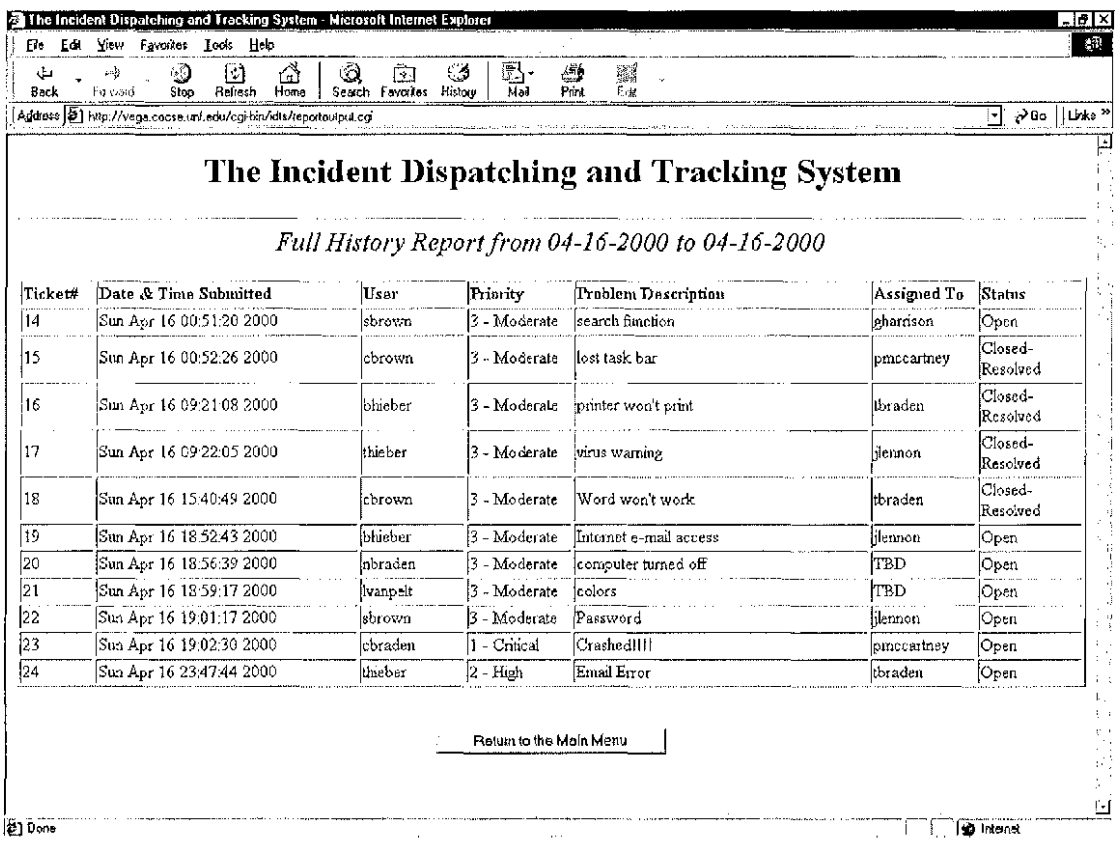
This report guide provides instructions on how to generate the following reports: Full History, Open Trouble Tickets, Trouble Ticket Audit Trail, Tickets Submitted by a Specific User, Commonly Reported Problems, and Keyword Search. The output is displayed in the browser for easy viewing or can be printed using the browser's print button.

Report Menu

From this form, select the desired report and click the Select Report button. This will display the report data form used to tailor the report and submit it for processing.

Full History Report

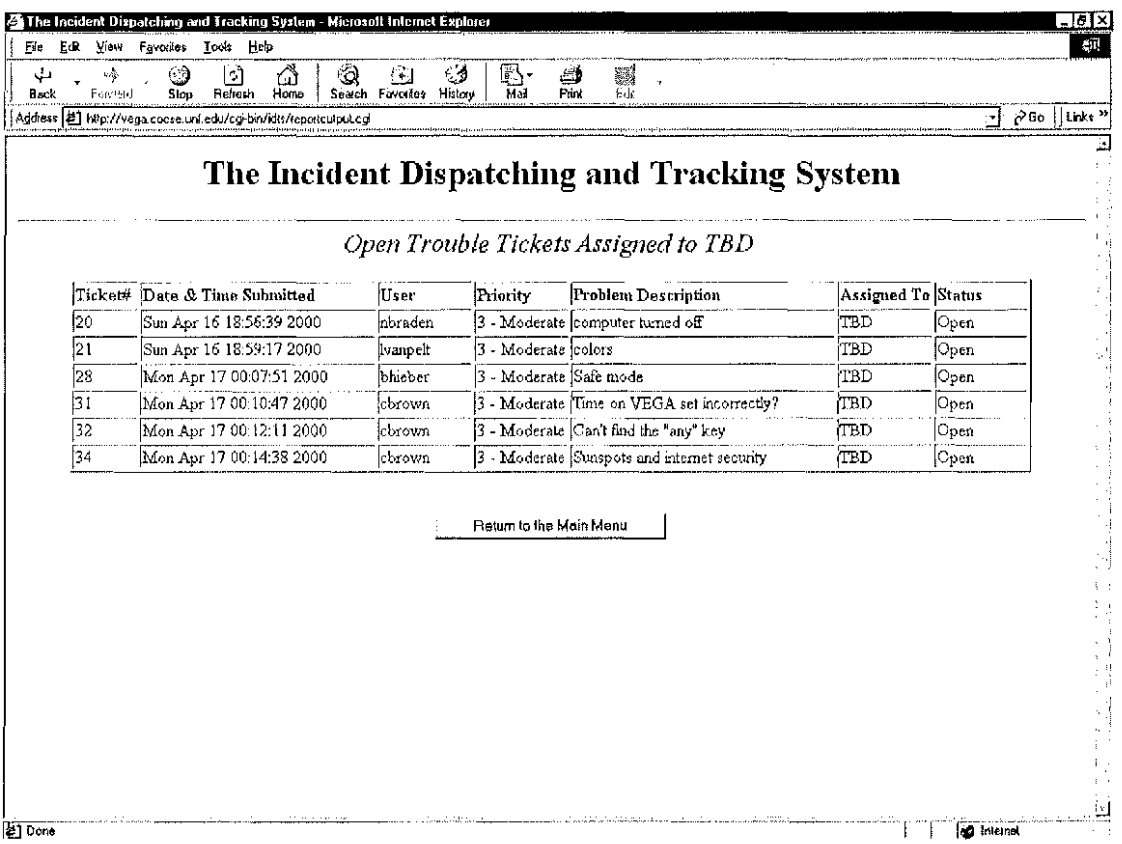
This report displays information for all tickets submitted within the requested date range and sorted by Username, Ticket# or Priority. To generate this report, enter the start date, end date, and sort criteria. The output will look like the following:



Ticket#	Date & Time Submitted	User	Priority	Problem Description	Assigned To	Status
14	Sun Apr 16 00:51:20 2000	sbrown	3 - Moderate	search function	gharrison	Open
15	Sun Apr 16 00:52:26 2000	cbrown	3 - Moderate	lost task bar	pmccartney	Closed-Resolved
16	Sun Apr 16 09:21:08 2000	bhieber	3 - Moderate	printer won't print	tbraden	Closed-Resolved
17	Sun Apr 16 09:22:05 2000	thieber	3 - Moderate	virus warning	jlennon	Closed-Resolved
18	Sun Apr 16 15:40:49 2000	cbrown	3 - Moderate	Word won't work	tbraden	Closed-Resolved
19	Sun Apr 16 18:52:43 2000	bhieber	3 - Moderate	Internet e-mail access	jlennon	Open
20	Sun Apr 16 18:56:39 2000	nbraden	3 - Moderate	computer turned off	TBD	Open
21	Sun Apr 16 18:59:17 2000	lvanpelt	3 - Moderate	colors	TBD	Open
22	Sun Apr 16 19:01:17 2000	sbrown	3 - Moderate	Password	jlennon	Open
23	Sun Apr 16 19:02:30 2000	cbraden	1 - Critical	Crashed!!!!	pmccartney	Open
24	Sun Apr 16 23:47:44 2000	thieber	2 - High	Email Error	tbraden	Open

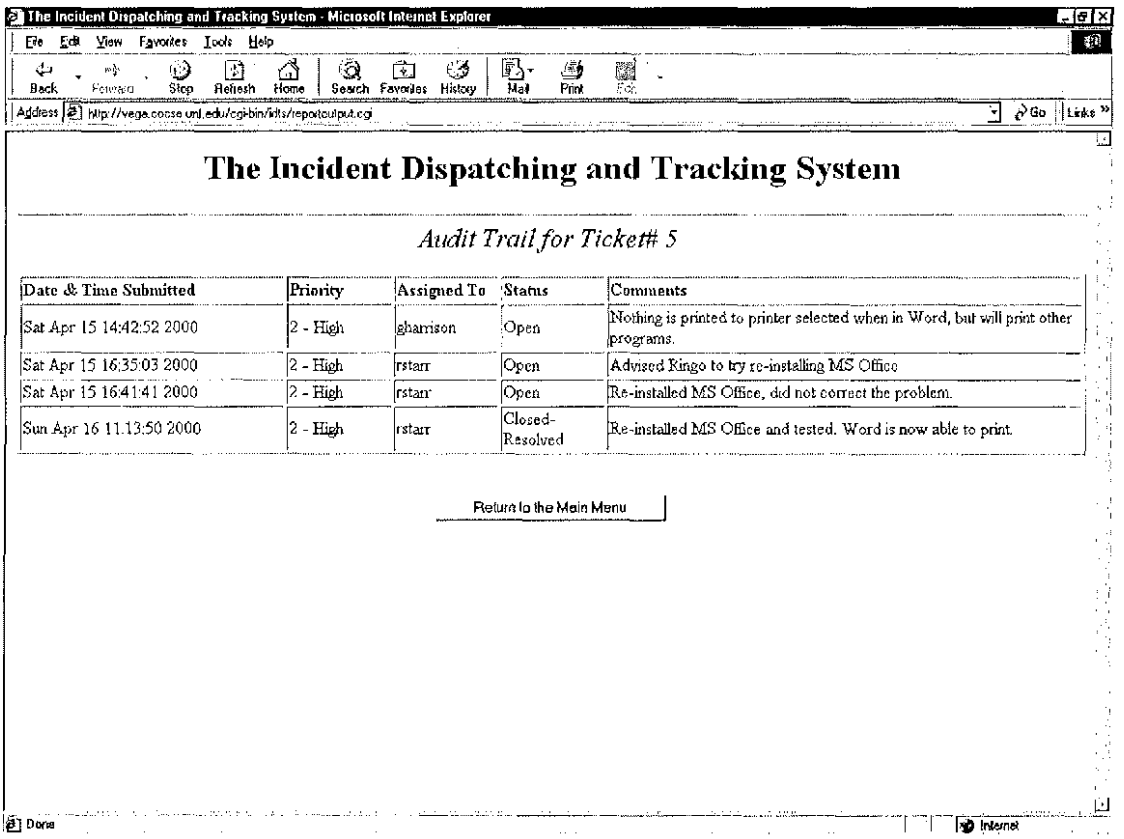
Open Trouble Tickets

This report displays information for either all open trouble tickets or those assigned to a specific technician and may be sorted by Username, Ticket# or Priority. To generate this report, select a technician or ALL and the sort criteria. This report is particularly useful for finding all tickets To Be Dispatched (TBD). The output will look like the following:



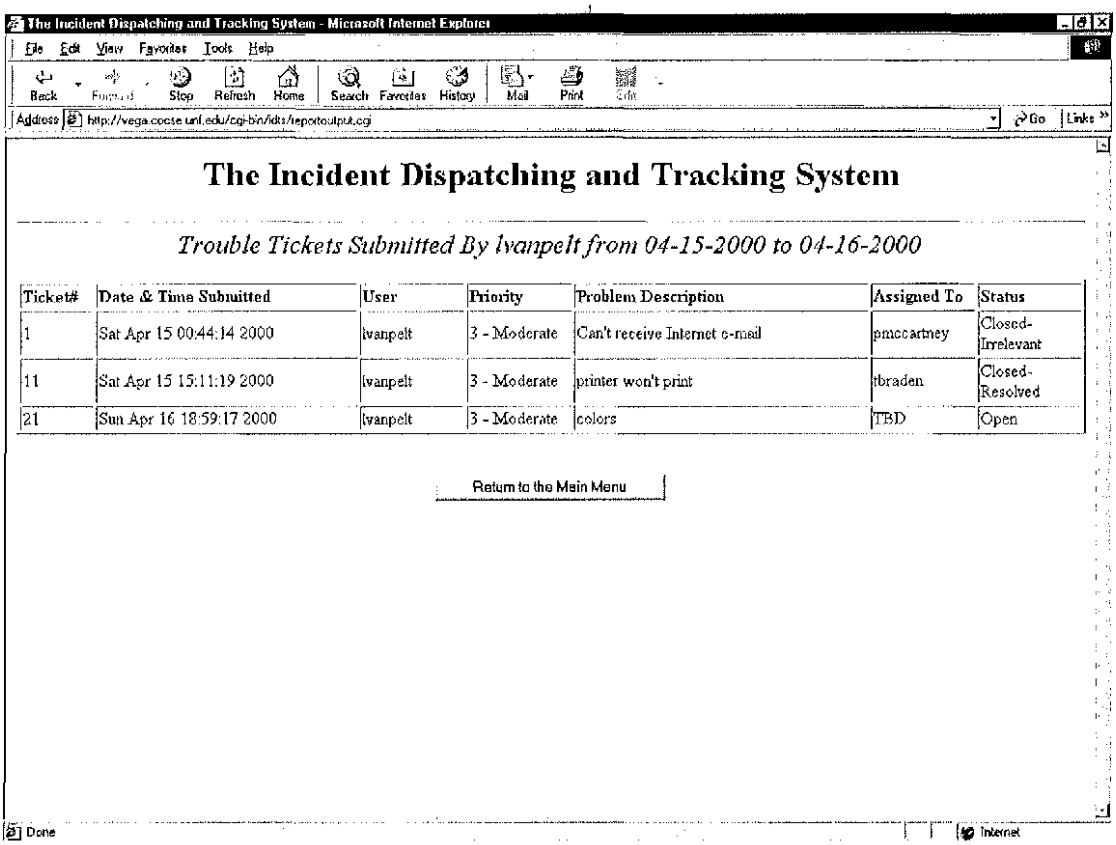
Trouble Ticket Audit Trail

This report displays the history for a requested trouble tickets. The output will look like the following:



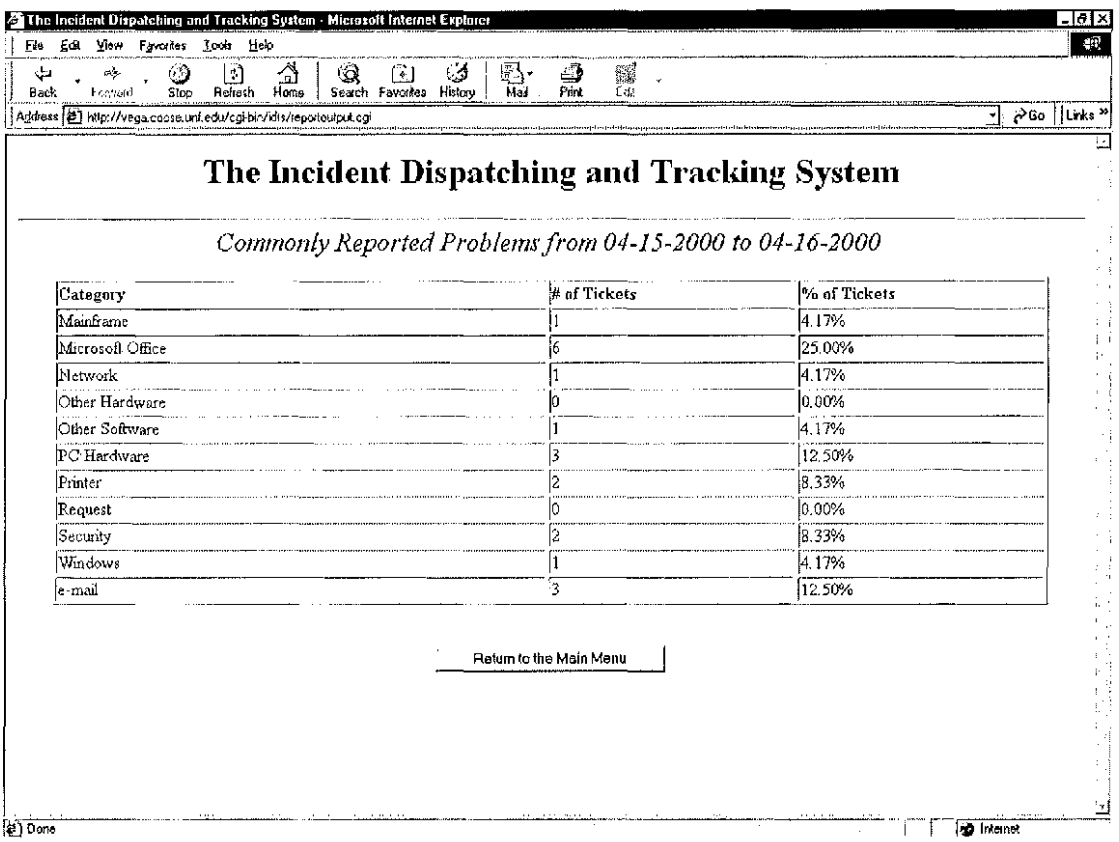
Tickets Submitted by a Specific User

This report displays information for all trouble tickets submitted by a specific user and may be sorted by Username, Ticket# or Priority. To generate this report, select a user and enter the start date and end date. The output will look like the following:



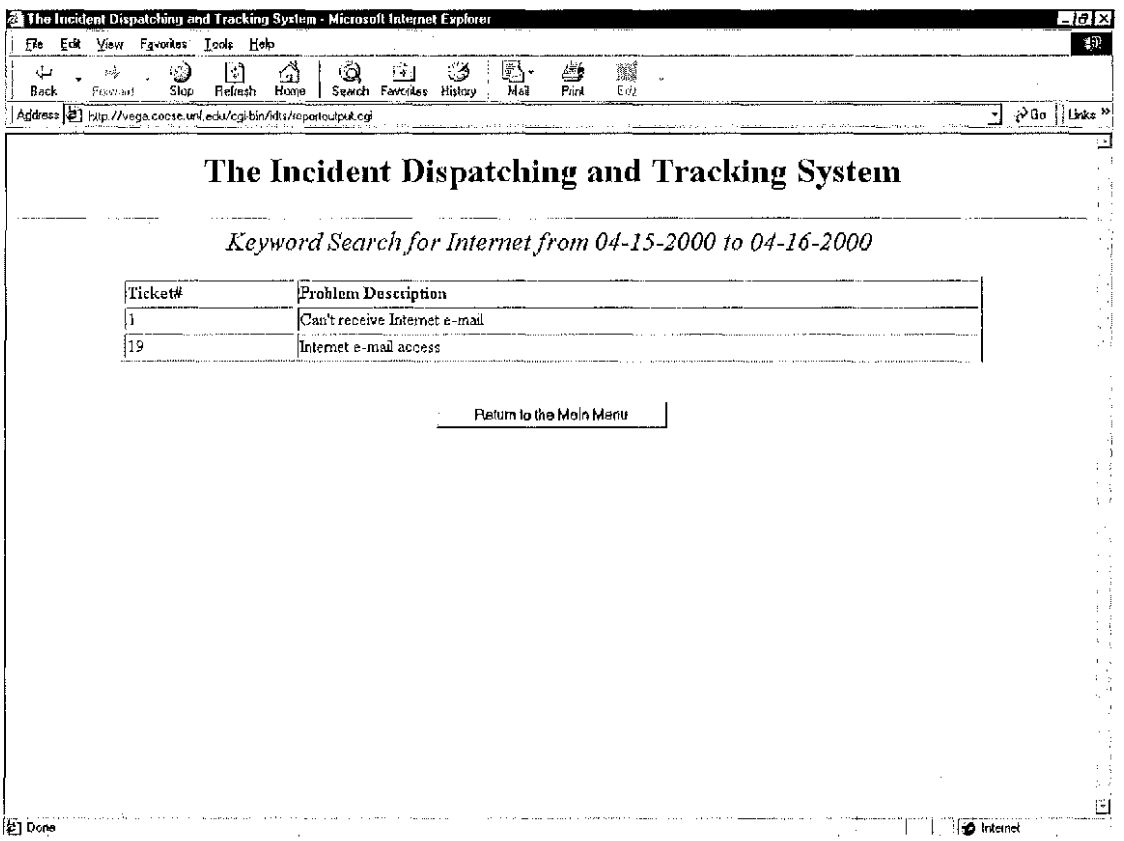
Commonly Reported Problems

This report displays statistics that detail the number of and percentage of trouble tickets for each category. To generate this report, enter the start date and end date. The output will look like the following:



Keyword Search

This report displays information for all trouble tickets containing any of the specified keywords. To generate this report, enter the start date, end date and one or more keywords separated by a space. The output will look like the following:



Appendix A - Source Code

adminadminconf.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to create or update Administrator or Technician access IDs.
#       This requires storing the information about the Admin/Tech in the ADMINS dbm
#       and the specialties of that Admin/Tech in teh ADMINSPEC dbm.
#
#*****************************************************************************
use CGI qw/:standard :netscape/;
use POSIX 'strftime';
require 'idts.lib';

%indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

#       Tying the ADMINS and ADMINSPEC dbms to hashes for use in creating and updating
#       Admin/Tech IDs
dbmopen(%admindata, "./ADMINS", 0644) || die "Can't access ADMINS, $!\n";
flock (ADMINS,2);
%temphash = %admindata;
dbmopen(%adminspecs, "./ADMINSPEC", 0644) || die "Can't access ADMINSPEC, $!\n";
flock (ADMINSPEC,2);

$adminspecs{$indata{'UserName'}} = $indata{'Specialties'};

if (($indata{'PasswordNew'} eq $indata{'Passwordchk'}) && ($indata{'Status'} eq
"Active")) {
        if ($indata{'UserName'} eq $indata{'ALoginID'}) {
                $indata{'Password'} = $indata{'PasswordNew'};
        }
        $value =
sprintf("%s~%s~%s~%s~%s~%s~%s~%s~%s~%s",$indata{'PasswordNew'},$indata{'FName'},$indata{'
LName'},$indata{'AdminPhone'},$indata{'PagerNo'},$indata{'AdminBldg'},$indata{'AdminRo
om'},$indata{'Specialties'},$indata{'RoleNew'});
        if (($indata{'admin'} eq "NEW") && (!exists ($temphash{$indata{'UserName'}})))
|| ($indata{'admin'} ne "NEW") && (exists ($temphash{$indata{'admin'}}))) {

#       Save Admin/Tech data for new and updated IDs
        $admindata{$indata{'UserName'}} = $value;
        dbmclose(%admindata);
        flock (ADMINS,8);
        dbmclose(%adminspecs);
        flock (ADMINSPEC,8);
        print center(br,i(font({-size=>5}, "Changes to Admin/Tech:
",$indata{'UserName'},," were submitted at",strftime('%I:%M %p on %A, %B %d
%Y',localtime),".")),
                start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                    hidden('ALoginID',$indata{'ALoginID'}),
                    hidden('Password',$indata{'Password'}),
                    hidden('Role',$indata{'Role'}),
                    br,
                    submit(-name=>'Return to the Main Menu'),
                end_form);
        } else {
```

```
#       Error - Attempting to create a new ID with an ID name that already exists
        dbmclose(%admindata);
        dbmclose(%adminspecs);
        print center(br,i(font({-size=>5}, "The Admin/Tech: ",$indata{'UserName'},"
already exists.")),
                start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
        }
} elsif ($indata{'Status'} eq "Delete") {

#       Delete the Admin/Tech ID
        delete($admindata{$indata{'UserName'}});
        delete($adminspecs{$indata{'UserName'}});
        dbmclose(%admindata);
        dbmclose(%adminspecs);
        print center(br,i(font({-size=>5}, "Admin/Tech:",$indata{'UserName'}," has
been deleted.")),
                start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
} else {

#       Error - New Password did not match retry
        dbmclose(%admindata);
        dbmclose(%adminspecs);
        print center(br,i(font({-size=>5}, "The Password check did not match.  Please
click Back and retype your passwords.")),
                start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
}

print end_html;
```

adminadmins.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to collect information to create or update Administrator or
#       Technician access IDs.  This requires collecting information about the
#       Admin/Tech from the form or retrieving information from the ADMINS dbm for
#       updating purposes.  This page is also used by Technicians to change their
#       password.
#
#*****************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

if (($indata{'admin'} eq 'NEW') && ($indata{'Role'} eq 'Administrator')) {
#       Provides a blank form for the creation of a new Admin/Tech

#       Read the CATEGORIES dbm to dynamically create a popup menu
        dbmopen(%catlist, "./CATEGORIES", 0644) || die "Can't access CATEGORIES,
$!\n";
        flock (CATEGORIES,2);
        %temphash = %catlist;
        dbmclose(%catlist);
        flock (CATEGORIES,8);
        print center(i(font({-size=>5}, "Create a New Administrator or Technician")),
                start_form(-method=>POST,
                        -action=>'adminadminconf.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                hidden('Status',"Active"),
                hidden('admin',$indata{'admin'}),
                br,
                "<TABLE>\n",
                "<TR><TD>Admin/Tech ID:</TD><TD>",
                textfield(-name=>'UserName',
                        -size=>30),"</TD></TR>",
                "<TR><TD>Password:</TD><TD>",
                password_field(-name=>'PasswordNew',
                        -size=>30),"</TD></TR>",
                "<TR><TD>Retype Password:</TD><TD>",
                password_field(-name=>'Passwordchk',
                        -size=>30),"</TD></TR>",
                "<TR><TD>Role:</TD><TD>",
                popup_menu(-name=>'RoleNew',
                        -value=>['Administrator','Technician'],
                        -default=>'Technician'),"</TD></TR>",
                "<TR><TD>First Name:</TD><TD>",
                textfield(-name=>'FName',
                        -size=>30),"</TD></TR>",
                "<TR><TD>Last Name:</TD><TD>",
                textfield(-name=>'LName',
                        -size=>30),"</TD></TR>",
                "<TR><TD>Phone:</TD><TD>",
                textfield(-name=>'AdminPhone',
                        -size=>30),"</TD></TR>",
```

```
                        "<TR><TD>Pager:</TD><TD>",
                        textfield(-name=>'PagerNo',
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Building:</TD><TD>",
                        textfield(-name=>'AdminBldg',
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Room:</TD><TD>",
                        textfield(-name=>'AdminRoom',
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Specialties:</TD><TD>",
                        scrolling_list('Specialties',
                                [sort keys(%temphash)],[@specs],3,'true'),
                                "</TD></TR>",
                        "</TABLE>",
                        br,
                        submit(-name=>'Submit',
                               -value=>'Submit Admin/Tech changes'),br,br,
                        reset(-name=>'Reset to Admin/Tech defaults'),
                        br,
                end_form);
} elsif (($indata{'admin'} ne 'NEW') && ($indata{'Role'} eq 'Administrator')) {
#       Retrieve data for an existing Admin/Tech for updating
        dbmopen(%admindata, "./ADMINS", undef) || die "Can't access ADMINS, $!\n";
        flock (ADMINS,2);
        @admintemp = split (/~/, $admindata{$indata{'admin'}});
        dbmclose(%admindata);
        flock (ADMINS,8);
        dbmopen(%catlist, "./CATEGORIES", 0644) || die "Can't access USERS, $!\n";
        flock (CATEGORIES,2);
        %temphash = %catlist;
        dbmclose(%catlist);
        flock(CATEGORIES,8);
        @specs = split (/@/, @admintemp[7]);
        print center(i(font({-size=>5}, "Submit changes for Admin/Tech:
",$indata{'admin'})),
                start_form(-method=>POST,
                    -action=>'adminadminconf.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        hidden('admin',$indata{'admin'}),
                        hidden('UserName',$indata{'admin'}),
                        br,
                        "<TABLE>\n",
                        "<TR><TD>Status:</TD><TD>",
                                popup_menu(-name=>'Status',
                                  -value=>[' ','Active','Delete'],
                                  -default=>'Active'),
                        "<TR><TD>Password:</TD><TD>",
                        password_field(-name=>'PasswordNew',
                                -value=>@admintemp[0],
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Retype Password:</TD><TD>",
                        password_field(-name=>'Passwordchk',
                                -value=>@admintemp[0],
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Role:</TD><TD>",
                        popup_menu(-name=>'RoleNew',
                                -value=>['Administrator','Technician'],
                                -default=>@admintemp[8]),
                        "<TR><TD>First Name:</TD><TD>",
                        textfield(-name=>'FName',
                                -value=>@admintemp[1],
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Last Name:</TD><TD>",
                        textfield(-name=>'LName',
                                -value=>@admintemp[2],
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Phone:</TD><TD>",
                        textfield(-name=>'AdminPhone',
                                -value=>@admintemp[3],
```

```
                        -size=>30),"</TD></TR>",
                    "<TR><TD>Pager:</TD><TD>",
                    textfield(-name=>'PagerNo',
                        -value=>@admintemp[4],
                        -size=>30),"</TD></TR>",
                    "<TR><TD>Building:</TD><TD>",
                    textfield(-name=>'AdminBldg',
                        -value=>@admintemp[5],
                        -size=>30),"</TD></TR>",
                    "<TR><TD>Room:</TD><TD>",
                    textfield(-name=>'AdminRoom',
                        -value=>@admintemp[6],
                        -size=>30),"</TD></TR>",
                    "<TR><TD>Specialties:</TD><TD>",
                    scrolling_list('Specialties',
                            [sort keys(%temphash)],[@specs],3,'true'),
                        "</TD></TR>",
                    "</TABLE>",
                    br,
                    submit(-name=>'Submit',
                        -value=>'Submit Admin/Tech changes'),br,br,
                    reset(-name=>'Reset to Admin/Tech defaults'),
                    br,
                end_form);
} elsif ($indata{'Role'} eq 'Technician') {

#       Technician password changing form
        dbmopen(%admindata, "./ADMINS", undef) || die "Can't access ADMINS, $!\n";
        flock (ADMINS,2);
        @admintemp = split (/~/, $admindata{$indata{'ALoginID'}});
        dbmclose(%admindata);
        flock (ADMINS,8);
        print center(i(font({-size=>5}, "Change Password for: ",$indata{'ALoginID'})),
                start_form(-method=>POST,
                        -action=>'adminadminconf.cgi'),
                    hidden('ALoginID',$indata{'ALoginID'}),
                    hidden('admin',$indata{'ALoginID'}),
                    hidden('Password',$indata{'Password'}),
                    hidden('Role',$indata{'Role'}),
                    hidden('Status',"Active"),
                    hidden('RoleNew',@admintemp[8]),
                    hidden('FName',@admintemp[1]),
                    hidden('LName',@admintemp[2]),
                    hidden('AdminPhone',@admintemp[3]),
                    hidden('PagerNo',@admintemp[4]),
                    hidden('AdminBldg',@admintemp[5]),
                    hidden('AdminRoom',@admintemp[6]),
                    hidden('Specialties',@admintemp[7]),
                    hidden('UserName',$indata{'ALoginID'}),
                    br,
                    "<TABLE>\n",
                    "<TR><TD>Password:</TD><TD>",
                    password_field(-name=>'PasswordNew',
                        -value=>@admintemp[0],
                        -size=>30),"</TD></TR>",
                    "<TR><TD>Retype Password:</TD><TD>",
                    password_field(-name=>'Passwordchk',
                        -value=>@admintemp[0],
                        -size=>30),"</TD></TR>",
                    "</TABLE>",
                    br,
                    submit(-name=>'Submit',
                        -value=>'Submit Password Change'),br,br,
                    reset(-name=>'Reset to Admin/Tech defaults'),
                    br,
                end_form);
}

print center(
        start_form(-action=>'adminmain.cgi'),
        hidden('ALoginID',$indata{'ALoginID'}),
```

```
        hidden('Password',$indata{'Password'}),
        hidden('Role',$indata{'Role'}),
        submit(-value=>'Return to the Main Menu'),
        end_form);

print end_html;
```

admincategory.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to create or delete trouble ticket categories.  These
#       categories are used to facilitate dispatching (by matching the ticket category
#       with tech specialties) and for reporting.
#
#*****************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
use NDBM_File;
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

if ($indata{'cat'} eq 'NEW') {

#       Form used to create a new category
        print center(i(font({-size=>5}, "Create a Category")),
                start_form(-method=>POST,
                        -action=>'admincategoryconf.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        hidden('cat',$indata{'cat'}),
                        br,
                        "<TABLE>",
                        "<TR><TD>New Category:</TD><TD>",
                        textfield(-name=>'NewCategory',
                                -size=>30),"</TD></TR>",
                        "</TABLE>",
                        br,
                        submit(-name=>'Submit',
                                -value=>'Submit New Category'),
                end_form);
} else {

#       Form used to edit an existing category
        dbmopen(%categorydata, "./CATEGORIES", undef) || die "Can't access CATEGORIES,
$!\n";
        flock (CATEGORIES,2);
        @categorytemp = split (/~/, $categorydata{$indata{'cat'}});
        dbmclose(%categorydata);
        flock (CATEGORIES,8);
        print center(i(font({-size=>5}, "Submit changes for Category:
",$indata{'cat'})),
                start_form(-method=>POST,
                        -action=>'admincategoryconf.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        hidden('NewCategory',$indata{'cat'}),
                        hidden('cat',$indata{'cat'}),
                        br,
                        "<TABLE>",
                        "<TR><TD>Status:</TD><TD>",
                                popup_menu(-name=>'Status',
                                  -value=>['Active','Delete'],
```

```
                                -default=>'Active'),
                        "</TABLE>",
                        br,
                        submit(-name=>'Submit',
                                -value=>'Submit Category Changes'),br,br,
                        reset(-name=>'Reset to Category defaults'),
                        br,
                end_form);
}

print center(
        start_form(-action=>'adminmain.cgi'),
        hidden('ALoginID',$indata{'ALoginID'}),
        hidden('Password',$indata{'Password'}),
        hidden('Role',$indata{'Role'}),
        submit(-value=>'Return to the Main Menu'),
        end_form);


print end_html;
```

admincategoryconf.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to create or delete Trouble Ticket categories.  Categorie
#       names are stored in the CATEGORIES dbm.
#
#*****************************************************************************
use CGI qw/:standard :netscape/;
use POSIX 'strftime';
require 'idts.lib';

%indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);


dbmopen(%categorydata, "./CATEGORIES", undef) || die "Can't access CATEGORIES, $!\n";
flock (CATEGORIES,2);
%temphash = %categorydata;

if (($indata{'cat'} eq "NEW") && (!exists ($temphash{$indata{'NewCategory'}}))) {

#       Adds the new category to the dbm if it doesn't already exist
        $categorydata{$indata{'NewCategory'}} = "";
        dbmclose (%categorydata);
        flock (CATEGORIES,8);

        print center(br,i(font({-size=>5}, "Category: ",$indata{'NewCategory'}," was
added at",strftime('%I:%M %p on %A, %B %d %Y',localtime),".")),
                start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                    hidden('ALoginID',$indata{'ALoginID'}),
                    hidden('Password',$indata{'Password'}),
                    hidden('Role',$indata{'Role'}),
                    br,
                    submit(-name=>'Return to the Main Menu'),
                end_form);
} elsif (($indata{'Status'} ne "Delete") && (exists
($temphash{$indata{'NewCategory'}}))) {

#       Error - Category name already exists
        dbmclose (%categorydata);
        flock (CATEGORIES,8);
        print center(br,i(font({-size=>5}, "Category: ",$indata{'NewCategory'},"
already exists.")),
                start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                    hidden('ALoginID',$indata{'ALoginID'}),
                    hidden('Password',$indata{'Password'}),
                    hidden('Role',$indata{'Role'}),
                    br,
                    submit(-name=>'Return to the Main Menu'),
                end_form);
} elsif (($indata{'Status'} eq "Delete") && (exists
($temphash{$indata{'NewCategory'}}))) {

#       Delete the category from the CATEGORIES dbm
        delete($categorydata{$indata{'NewCategory'}});
        dbmclose (%categorydata);
        flock (CATEGORIES,8);
```

```
        print center(br,i(font({-size=>5}, "The Category: ",$indata{'NewCategory'},"
has been deleted.")),
                start_form(-method=>POST,
                          -action=>'adminmain.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
} elsif (($indata{'Status'} eq "Active")) {

#       No action taken
        dbmclose (%categorydata);
        flock (CATEGORIES,8);
        print center(br,i(font({-size=>5}, "No action taken on Category:
",$indata{'NewCategory'})),
                start_form(-method=>POST,
                          -action=>'adminmain.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
} else {

#       Record does not exist so it can't be deleted
        dbmclose (%categorydata);
        flock (CATEGORIES,8);
        print center(br,i(font({-size=>5}, "The Category: ",$indata{'NewCategory'},"
cannot be deleted.")),
                start_form(-method=>POST,
                          -action=>'adminmain.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
}

print end_html;
```

adminlogin.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to collect the Admin's/Tech's ID and password for login.
#
#*****************************************************************************
use CGI qw/:standard :netscape/;

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,'System Login
Page'),
                hr);


print center(i(font({-size=>5}, "Please enter your ID to Log on to the system.")),
        start_form(-method=>POST,
                   -action=>'adminmain.cgi'),
                "<TABLE>",
                "<TR><TD>Username:</TD><TD>",
                textfield('ALoginID'),"</TD></TR>",
                "<TR><TD>Password:</TD><TD>",
                password_field('Password'),"</TD></TR>",
                "</TABLE>",br,
                submit(-name=>'Submit',
                        -value=>'LogOn'),
                br,
        end_form);

print end_html;
```

adminmain.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to display the main menu for an Administrator or a
#       Technician.
#
#*****************************************************************************
use CGI qw/:standard :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

#       Tying the ADMINS dbm to a hash for use in identifying the role and verifying
the password
dbmopen(%admindata, "./ADMINS", 0644) || die "Can't access ADMINS, $!\n";
flock (ADMINS,2);
%temphash = %admindata;
@admintemp = split (/~/, $admindata{$indata{'ALoginID'}});
dbmclose(%admindata);
flock (ADMINS,8);

if (exists ($temphash{$indata{'ALoginID'}}) && ($indata{'Password'} eq @admintemp[0])
&& (@admintemp[8] eq "Administrator")) {

#       Display the Administrator's Main Menu
        print center(
                start_form,
                "Welcome, ",@admintemp[1]," ",@admintemp[2],
                br,
                end_form);
        print center(
                start_form,
                font({-size=>5}, "Please make your selection below"),
                br,
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'lookuptickets.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Lookup a Trouble Ticket'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'ticketselectstatus.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Check a Trouble Ticket'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'adminselectticket.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Role',@admintemp[8]),
                hidden('Password',$indata{'Password'}),
```

```
                submit(-value=>'Modify an existing ticket'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'adminselectcategory.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Administer Categories'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'adminselectuser.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Administer a User'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'adminselectadmin.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Administer an Admin/Tech'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'reports.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Run a Report'),
                end_form);
} elsif (exists ($temphash{$indata{'ALoginID'}}) && ($indata{'Password'} eq
@admintemp[0]) && @admintemp[8] eq "Technician") {

#       Display the Technician's Main Menu
        print center(
                start_form,
                "Welcome, ",@admintemp[1]," ",@admintemp[2],
                br,
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'lookuptickets.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Lookup a Trouble Ticket'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'ticketselectstatus.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Check a Trouble Ticket'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'adminselectticket.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Modify an existing ticket'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'reports.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
```

```
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Run a Report'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'adminadmins.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',@admintemp[8]),
                submit(-value=>'Change Password'),
                end_form);
} else {

#       Error - Username invalid or incorrect password
        print center(
                start_form,
                "Username or Password is incorrect",
                br,
                end_form);
}

print center(
        start_form(-action=>'adminlogin.cgi'),
        submit(-value=>'Logout'),
        end_form);

print end_html;
```

adminselectadmin.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to select the Administrator or Technician to be updated or
#       NEW to create a new Admin/Tech.  This page uses the ADMINS dbm to dynamically
#       create the popup menu of choices.
#
#*****************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,
        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

#       Tying the ADMINS dbm to a hash for use in dynamically creating a popup menu
dbmopen(%adminlist, "./ADMINS", 0644) || die "Can't access ADMINS, $!\n";
flock (ADMINS,2);
%temphash = %adminlist;
dbmclose(%adminlist);
flock (ADMINS,8);

print center(i(font({-size=>5}, "Please select an Administrator or Technician.")),
        start_form(-method=>POST,
                -action=>'adminadmins.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                br,
#       create popup menu of currently existing Admins/Techs
                popup_menu(-name=>'admin',
                        -value=>['NEW',sort keys(%temphash)],
                        -default=>' '),
                br,br,
                submit(-name=>'Submit',
                        -value=>'Access Admin/Tech Information'),
                br,
        end_form);

print center(
        start_form(-action=>'adminmain.cgi'),
        hidden('ALoginID',$indata{'ALoginID'}),
        hidden('Password',$indata{'Password'}),
        hidden('Role',$indata{'Role'}),
        submit(-value=>'Return to the Main Menu'),
        end_form);

print end_html;
```

adminselectcategory.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to select the Category to be updated or NEW to create a new
#       Category.  This page uses the CATEGORIES dbm to dynamically create the popup
#       menu of choices.
#
#*****************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

#       Tying the CATEGORIES dbm to a hash for use in dynamically creating a popup
menu
dbmopen(%categorydata, "./CATEGORIES", 0644) || die "Can't access CATEGORIES, $!\n";
flock(CATEGORIES,2);
%temphash = %categorydata;
dbmclose(%categorydata);
flock(CATEGORIES,8);

print center(i(font({-size=>5}, "Please select a category.")),
        start_form(-method=>POST,
                   -action=>'admincategory.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                br,
                popup_menu(-name=>'cat',
                          -value=>['NEW',sort keys(%temphash)],
                          -default=>'NEW'),
                br,br,
                submit(-name=>'Submit',
                       -value=>'Access Category Information'),
                br,
        end_form);

print center(
        start_form(-action=>'adminmain.cgi'),
        hidden('ALoginID',$indata{'ALoginID'}),
        hidden('Password',$indata{'Password'}),
        hidden('Role',$indata{'Role'}),
        submit(-value=>'Return to the Main Menu'),
        end_form);

print end_html;
```

adminselectticket.cgi

```
#!/usr/bin/perl
#*********************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to request the Trouble Ticket number to be updated.
#
#*********************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
        hr);

print center(i(font({-size=>5}, "Please enter the Trouble Ticket # that you would like
to review.")),
        start_form(-method=>POST,
                  -action=>'adminticket.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                br,
                textfield(-name=>'Ticket',
                  -size=>11),
                br,br,
                submit(-name=>'Submit',
                        -value=>'Modify this Ticket'),
                br,
        end_form);


print center(
        start_form(-method=>POST,
                  -action=>'adminmain.cgi'),
        hidden('ALoginID',$indata{'ALoginID'}),
        hidden('Password',$indata{'Password'}),
        hidden('Role',$indata{'Role'}),
        submit(-value=>'Return to the Main Menu'),
        end_form);


print end_html;
```

adminselectuser.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to select the User ID to be updated or NEW to     create a
#       new User ID.  This page uses the USERS dbm to dynamically create the popup
#       menu of choices.
#
#*****************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,
        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

#       Tying the USERS dbm to a hash for use in dynamically creating a popup menu
dbmopen(%userlist, "./USERS", 0644) || die "Can't access USERS, $!\n";
flock(USERS,2);
%temphash = %userlist;
dbmclose(%userlist);
flock(USERS,8);

print center(i(font({-size=>5}, "Please select a user.")),
        start_form(-method=>POST,
                -action=>'adminuser.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                br,
                popup_menu(-name=>'user',
                        -value=>['NEW',sort keys(%temphash)],
                        -default=>'NEW'),
                br,br,
                submit(-name=>'Submit',
                        -value=>'Access User Information'),
                br,
        end_form);

print center(
        start_form(-action=>'adminmain.cgi'),
        hidden('ALoginID',$indata{'ALoginID'}),
        hidden('Password',$indata{'Password'}),
        hidden('Role',$indata{'Role'}),
        submit(-value=>'Return to the Main Menu'),
        end_form);

print end_html;
```

adminticket.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to collect information to update a Trouble Ticket.  This
#       requires collecting information about the Trouble Ticket from the TICKETS dbm.
#       All instances of the ticket are collected and the current data is displayed
#       along with all comments that were previously entered.
#
#*****************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

#       Read the TICKETS dbm and search for all records for the requested Trouble
Ticket
$ticketnum = $indata{'Ticket'};
dbmopen(%ticketdata, "./TICKETS", 0644) || die "Can't access TICKETS, $!\n";
flock(TICKETS,2);
for (keys(%ticketdata)) {
        if ((sprintf("%s",$_ =~ m/^([\d]+)/)) eq $ticketnum) {
                push (@tktmodify, $_);
        }
}
@tktmodify = sort @tktmodify;

#       Collect the comments for the requested ticket in order
@keytemp = split (/~/, @tktmodify[0]);
@valuetemp = split (/~/, $ticketdata{@tktmodify[(scalar(@tktmodify)-1)]});
$comments = "";
for (@tktmodify) {
        @commenttemp = split (/~/, $ticketdata{$_});
        if (@commenttemp[6] ne "") {
                $comments .= @commenttemp[6] . "\n\n";
        }
}

dbmclose (%ticketdata);
flock(TICKETS,8);

#       Read the ADMINS dbm to dynamically create a popup menu
dbmopen(%admintemp, "./ADMINS", 0644) || die "Can't access ADMINS, $!\n";
flock(ADMINS,2);
        @admins = sort keys(%admintemp);
dbmclose(%admintemp);
flock(ADMINS,8);

#       Read the CATEGORIES dbm to dynamically create a popup menu
dbmopen(%categorydata, "./CATEGORIES", 0644) || die "Can't access CATEGORIES, $!\n";
flock(CATEGORIES,2);
%temphash = %categorydata;
dbmclose(%categorydata);
flock(CATEGORIES,8);

#       Create the form containing the Trouble Ticket's current data
print center(i(font({-size=>5}, "Ticket Number:",sprintf("%s,
%s",$ticketnum,@valuetemp[0]))),
```

```
        start_form(-method=>POST,
                   -action=>'ticketsubmitconf.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('LoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                hidden('Ticket',$indata{'Ticket'}),
                hidden('tktnum',@keytemp[0]),
                hidden('ProbBrief',@valuetemp[0]),
                hidden('UserPhone',@valuetemp[1]),
                hidden('UserBldg',@valuetemp[2]),
                hidden('UserRoom',@valuetemp[3]),
                br,
                "<TABLE>\n",
                "<TR><TD>UserName:</TD><TD>",@keytemp[2],"</TD></TR>",
                "<TR><TD>Phone:</TD><TD>",@valuetemp[1],"</TD></TR>",
                "<TR><TD>Building:</TD><TD>",@valuetemp[2],"</TD></TR>",
                "<TR><TD>Room:</TD><TD>",@valuetemp[3],"</TD></TR>",
                "<TR><TD>Category:</TD><TD>",
                popup_menu(-name=>'TroubleCategory',
                           -value=>['Not Sure',sort keys(%temphash)],
                           -default=>@valuetemp[4]),"</TD></TR>",
                "<TR><TD>Re-Dispatch?:</TD><TD>",
                popup_menu(-name=>'ReDispatch',
                           -value=>['Yes','No'],
                           -default=>'No'),"</TD></TR>",
                "<TR><TD>Previous Comments:</TD><TD>",
                textarea(-name=>'PrevComments',
                           -value=>$comments,
                           -rows=>5,
                           -columns=>40,
                           -wrap=>'physical'),"</TD></TR>",
                "<TR><TD>Additional Comments:</TD><TD>",
                textarea(-name=>'TroubleDescription',
                           -value=>'',
                           -rows=>5,
                           -columns=>40,
                           -wrap=>'physical'),"</TD></TR>",
                "<TR><TD>Priority:</TD><TD>",
                popup_menu(-name=>'TroublePriority',
                           -value=>['1 - Critical','2 - High','3 - Moderate','4 -
Low','5 - Very Low'],
                           -default=>@valuetemp[7]),"</TD></TR>",
                "<TR><TD>Assigned to:</TD><TD>",
                popup_menu(-name=>'AssignedTo',
                           -value=>['TBD',@admins],
                           -default=>@valuetemp[5]),"</TD></TR>",
                "<TR><TD>Status:</TD><TD>",
                popup_menu(-name=>'Status',
                           -value=>['Open','Suspended','Closed-Resolved','Closed-
Unsolvable','Closed-Irrelevant'],
                           -default=>@valuetemp[8]),"</TD></TR>",
                "</TABLE>",
                br,
                submit(-name=>'Submit',
                       -value=>'Submit changes to this Ticket'),br,br,
                reset(-name=>'Reset this ticket'),
                br,
        end_form);

print center(
        start_form(-method=>POST,
                   -action=>'adminmain.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                submit(-value=>'Return to the Main Menu'),
        end_form);

print end_html;
```

adminuser.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to collect information to create or update a User's access
#       IDs.  This requires collecting information about the User from the form or
#       retrieving information from the USERS dbm for updating purposes.
#
#*****************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
use NDBM_File;
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

if ($indata{'user'} eq 'NEW') {
#       Provides a blank form for the creation of a new User
        print center(i(font({-size=>5}, "Create a New User")),
                start_form(-method=>POST,
                        -action=>'adminuserconf.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        hidden('Status',"Active"),
                        hidden('user',$indata{'user'}),
                        br,
                        "<TABLE>",
                        "<TR><TD>Username:</TD><TD>",
                        textfield(-name=>'UserName',
                                -size=>30),"</TD></TR>",
                        "<TR><TD>First Name:</TD><TD>",
                        textfield(-name=>'FName',
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Last Name:</TD><TD>",
                        textfield(-name=>'LName',
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Phone:</TD><TD>",
                        textfield(-name=>'UserPhone',
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Building:</TD><TD>",
                        textfield(-name=>'UserBldg',
                                -size=>30),"</TD></TR>",
                        "<TR><TD>Room:</TD><TD>",
                        textfield(-name=>'UserRoom',
                                -size=>30),"</TD></TR>",
                        "</TABLE>",
                        br,
                        submit(-name=>'Submit',
                                -value=>'Submit user changes'),br,br,
                        reset(-name=>'Reset to user defaults'),
                        br,
                end_form);
} else {
#       Retrieve data for an existing User for updating
        dbmopen(%userdata, "./USERS", undef) || die "Can't access USERS, $!\n";
        flock(USERS,2);
        @usertemp = split (/~/, $userdata{$indata{'user'}});
        dbmclose(%userdata);
```

```
        flock(USERS,8);
        print center(i(font({-size=>5}, "Submit changes for username:
",$indata{'user'})),
                start_form(-method=>POST,
                        -action=>'adminuserconf.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        hidden('UserName',$indata{'user'}),
                        hidden('user',$indata{'user'}),
                        br,
                        "<TABLE>",
                        "<TR><TD>Status:</TD><TD>",
                                popup_menu(-name=>'Status',
                                  -value=>[' ','Active','Delete'],
                                  -default=>'Active'),
                        "<TR><TD>First Name:</TD><TD>",
                        textfield(-name=>'FName',
                                  -value=>@usertemp[0],
                                  -size=>30),"</TD></TR>",
                        "<TR><TD>Last Name:</TD><TD>",
                        textfield(-name=>'LName',
                                  -value=>@usertemp[1],
                                  -size=>30),"</TD></TR>",
                        "<TR><TD>Phone:</TD><TD>",
                        textfield(-name=>'UserPhone',
                                  -value=>@usertemp[2],
                                  -size=>30),"</TD></TR>",
                        "<TR><TD>Building:</TD><TD>",
                        textfield(-name=>'UserBldg',
                                  -value=>@usertemp[3],
                                  -size=>30),"</TD></TR>",
                        "<TR><TD>Room:</TD><TD>",
                        textfield(-name=>'UserRoom',
                                  -value=>@usertemp[4],
                                  -size=>30),"</TD></TR>",
                        "</TABLE>",
                        br,
                        submit(-name=>'Submit',
                                -value=>'Submit user changes'),br,br,
                        reset(-name=>'Reset to user defaults'),
                        br,
                end_form);
}

print center(
        start_form(-action=>'adminmain.cgi'),
        hidden('ALoginID',$indata{'ALoginID'}),
        hidden('Password',$indata{'Password'}),
        hidden('Role',$indata{'Role'}),
        submit(-value=>'Return to the Main Menu'),
        end_form);

print end_html;
```

adminuserconf.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to create or update a User access ID.  This requires storing
#       the information about the User in the USERS dbm.
#
#*****************************************************************************
use CGI qw/:standard :netscape/;
use POSIX 'strftime';
require 'idts.lib';

%indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

#       Tying the USERS dbm to a hash for use in creating and updating User ID
dbmopen(%userdata, "./USERS", undef) || die "Can't access USERS, $!\n";
flock(USERS,2);
%temphash = %userdata;

if (($indata{'Status'} eq "Active") && ((($indata{'user'} eq "NEW") && (!exists
($temphash{$indata{'UserName'}}))) || (($indata{'user'} ne "NEW") && (exists
($temphash{$indata{'UserName'}})))))) {

#       Create a new user or update an existing user ID
        $userdata{$indata{'UserName'}} =
sprintf("%s~%s~%s~%s~%s",$indata{'FName'},$indata{'LName'},$indata{'UserPhone'},$indat
a{'UserBldg'},$indata{'UserRoom'});
        dbmclose (%userdata);
        flock(USERS,8);
        print center(br,i(font({-size=>5}, "Changes to username:
",$indata{'UserName'}," were submitted at",strftime('%I:%M %p on %A, %B %d
%Y',localtime),".")),
                start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
} elsif (($indata{'Status'} eq "Delete") && (exists ($temphash{$indata{'UserName'}})))
{

#       Delete an existing user ID
        delete($userdata{$indata{'UserName'}});
        dbmclose (%userdata);
        flock(USERS,8);
        print center(br,i(font({-size=>5}, "The username: ",$indata{'UserName'}," has
been deleted.")),
                start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
} else {
```

```
#       Error - Attempting to create a user ID that already exists
        dbmclose (%userdata);
        flock(USERS,8);
        print center(br,i(font({-size=>5}, "The username: ",$indata{'UserName'},"
already exists.")),
                start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
}

print end_html;
```

idts.lib

```
#*********************************************************************************************
#
#       Tom Braden      April 2000
#
#       This lib file contains subroutines that are used in multiple pages.  Each page
#       requires idts.lib and uses at least one of these subroutines.
#
#*********************************************************************************************

sub getcgivars {
#*********************************************************************************************
#       This subroutine is used to retrieve the POSTed data from the previous page and
#       split it into a hash containing name/value pairs.  The hash is the return
#       value.
#
#       This subroutine was obtained and slightly modified from the example given by
#       James Marshal at http://www.jmarshall.com/easy/cgi/
#*********************************************************************************************
local($in, %in) ;
local($name, $value) ;


# First, read entire string of CGI vars into $in

        read(STDIN, $in, $ENV{'CONTENT_LENGTH'});

        $specialties = "";
# Resolve and unencode name/value pairs into %in
        foreach (split('&', $in)) {
                s/\+/ /g ;
                ($name, $value)= split('=', $_, 2) ;
                $name=~ s/%(..)/chr(hex($1))/ge ;
                $value=~ s/%(..)/chr(hex($1))/ge ;
                $in{$name}.= "@" if defined($in{$name}) ;  # concatenate multiple vars
                $in{$name}.= $value ;
        }

        return %in ;

}

sub printreport {
#*********************************************************************************************
#       This subroutine takes a hash (%report) and displays it to the browser as a row
#       in a table.  The key of %hash is the item to be sorted on and the value
#       contains the data seperated by ~s.  Each value is displayed as a new table row
#       and each piece of data is displayed as an item in that row.
#*********************************************************************************************
local @keys, @reportline;

        @keys = keys(%report);
#       sorts the output based on the key of the hash
        foreach (sort @keys) {
#       splits each value into its components for display as a table definition
                @reportline = split (/~/, $report{$_});
                print "<TR>";
                foreach $reportitem (@reportline) {
                        print "<TD>",$reportitem,"</TD>";
                }
                print "</TR>";
        }
        print "</TABLE>",
        end_form;
}

1;      #       return value for successful completion
```

lookuptickets.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to list ticket numbers and descriptions for use with other
#       pages that require ticket numbers as input.
#
#*****************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
        hr);


print center(i(font({-size=>5}, "Open Trouble Tickets")),
        start_form,
                "<TABLE border=1>",
                "<TH WIDTH=7% ALIGN=LEFT><B>Ticket#</B></TH><TH WIDTH=25%
ALIGN=LEFT><B>Date & Time Submitted</B></TH><TH WIDTH=10%
ALIGN=LEFT><B>Priority</B></TH><TH WIDTH=28% ALIGN=LEFT><B>Problem
Description</B></TH><TH WIDTH=10% ALIGN=LEFT><B>Assigned To</B></TH>");
                $DBMFILE = "./OPEN";
                ticketlookup();
                print br,br;

print center(i(font({-size=>5}, "All Trouble Tickets")),
        start_form,
                "<TABLE border=1>",
                "<TH WIDTH=7% ALIGN=LEFT><B>Ticket#</B></TH><TH WIDTH=25%
ALIGN=LEFT><B>Date & Time Submitted</B></TH><TH WIDTH=28% ALIGN=LEFT><B>Problem
Description</B></TH>");
                $DBMFILE = "./TICKETINDEX";
                if ($indata{'Role'} eq "Technician") {
                        $indata{'Role'} = "Administrator";
                }
                ticketlookup();


if ($indata{'LoginID'} ne "") {
        print center(
                start_form(-method=>POST,
                        -action=>'usermain.cgi'),
                hidden('LoginID',$indata{'LoginID'}),
                submit(-value=>'Return to the Main Menu'),
                end_form);
} elsif ($indata{'ALoginID'} ne "") {
        print center(
                start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                submit(-value=>'Return to the Main Menu'),
                end_form);
}
```

```
sub ticketlookup {
#*****************************************************************************************
#       This subroutine is used to look up the Date & Time that the ticket was
#       submitted, the Problem Description and who is assigned to any Open ticket.
#*****************************************************************************************
local @keytemp, @valuetemp;
local %report;

        dbmopen(%ticketdata, "./TICKETS", 0644) || die "Can't access TICKETS, $!\n";
        flock(TICKETS,2);
        dbmopen(%temptickets, $DBMFILE, 0644) || die "Can't access OPEN, $!\n";
        flock($DBMFILE,3);
        for (keys(%temptickets)) {
                if ((($_ =~ /$indata{'LoginID'}/) && ($indata{'Role'} eq "")) ||
(($temptickets{$_} =~ /$indata{'ALoginID'}/) && ($indata{'Role'} eq "Technician")) ||
($indata{'Role'} eq "Administrator")) {
                        @keytemp = split (/~/,$_);
                        @valuetemp = split (/~/,$ticketdata{$_});
                        if ($DBMFILE eq "./OPEN") {
                                $report{(sprintf("%08d",@keytemp[0]))} =
sprintf("%s~%s~%s~%s~%s",@keytemp[0],strftime('%c',localtime(@keytemp[1])),@valuetemp[
7],@valuetemp[0],$temptickets{$_});
                        } elsif ($DBMFILE eq "./TICKETINDEX") {
                                $report{(sprintf("%08d",@keytemp[0]))} =
sprintf("%s~%s~%s",@keytemp[0],strftime('%c',localtime(@keytemp[1])),@valuetemp[0]);
                        }
                }
        }
        dbmclose(%ticketdata);
        flock(TICKETS,8);
        dbmclose(%temptickets);
        flock($DBMFILE,8);
        printreport();

}

print end_html;
```

reportdata.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to collect information to produce the requested report.
#
#*****************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

if ($indata{'report'} eq 'Full History') {
        print center(i(font({-size=>5}, $indata{'report'}, " Report")),
                start_form(-method=>POST,
                        -action=>'reportoutput.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Role',$indata{'Role'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('DBMFILE',"./TICKETINDEX"),
                        hidden('report',$indata{'report'}),
                        br,
                        "<TABLE>",
                        "<TR><TD>Start Date</TD><TD>",
                        textfield(-name=>'startdate',
                                -value=>'MM-DD-YYYY',
                                -size=>11),"</TD></TR>",
                        "<TR><TD>End Date</TD><TD>",
                        textfield(-name=>'enddate',
                                -value=>'MM-DD-YYYY',
                                -size=>11),"</TD></TR>",
                        "<TR><TD>Sort By</TD><TD>",
                        popup_menu(-name=>'Sort',
                                -value=>['User','Ticket#','Priority'],
                                -default=>'Ticket#'),"</TD></TR>",
                        "</TABLE>",br,
                        submit(-name=>'Submit',
                                -value=>'Produce Report'),
                        br,
                end_form);
} elsif ($indata{'report'} eq 'Open Trouble Tickets') {
#       Read the ADMINS dbm to dynamically create a popup menu
        dbmopen(%admintemp, "./ADMINS", 0644) || die "Can't access ADMINS, $!\n";
        flock(ADMINS,2);
                @admins = sort keys(%admintemp);
        dbmclose(%admintemp);
        flock(ADMINS,8);
        print center(i(font({-size=>5}, $indata{'report'}, " Report")),
                start_form(-method=>POST,
                        -action=>'reportoutput.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Role',$indata{'Role'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('DBMFILE',"./OPEN"),
                        hidden('startdate',"00-00-0000"),
                        hidden('enddate',"99-99-9999"),
                        hidden('report',$indata{'report'}),
                        br,
```

```
                    "<TABLE>",
                    "<TR><TD>Select a Technician</TD><TD>",
                    popup_menu(-name=>'AssignedTo',
                            -value=>['All','TBD',@admins],
                            -default=>'All'),"</TD></TR>",
                    "<TR><TD>Sort by</TD><TD>",
                    popup_menu(-name=>'Sort',
                            -value=>['User','Ticket#','Priority'],
                            -default=>'Ticket#'),"</TD></TR></TABLE>",
                    br,
                    submit(-name=>'Submit',
                           -value=>'Produce Report'),
                    br,
                end_form);
} elsif ($indata{'report'} eq 'Tickets submitted by a specific User') {
#       Read the USERS dbm to dynamically create a popup menu
        dbmopen(%usertemp, "./USERS", 0644) || die "Can't access USERS, $!\n";
        flock(USERS,2);
        @users = sort keys(%usertemp);
        dbmclose(%usertemp);
        flock(USERS,8);
        print center(i(font({-size=>5}, $indata{'report'}, " Report")),
                start_form(-method=>POST,
                           -action=>'reportoutput.cgi'),
                    hidden('ALoginID',$indata{'ALoginID'}),
                    hidden('Role',$indata{'Role'}),
                    hidden('Password',$indata{'Password'}),
                    hidden('DBMFILE',"./TICKETINDEX"),
                    hidden('Sort',"Ticket#"),
                    hidden('report',$indata{'report'}),
                    "<TABLE>",
                    "<TR><TD>Start Date</TD><TD>",
                    textfield(-name=>'startdate',
                            -value=>'MM-DD-YYYY',
                            -size=>11),"</TD></TR>",
                    "<TR><TD>End Date</TD><TD>",
                    textfield(-name=>'enddate',
                            -value=>'MM-DD-YYYY',
                            -size=>11),"</TD></TR>",
                    "<TR><TD>Select a User</TD><TD>",
                    popup_menu(-name=>'UserName',
                            -value=>[@users]),"</TD></TR>",
                    "</TABLE>",br,
                    submit(-name=>'Submit',
                           -value=>'Produce Report'),
                end_form);
} elsif ($indata{'report'} eq 'Trouble Ticket Audit Trail') {
        print center(i(font({-size=>5}, $indata{'report'}, " Report")),
                start_form(-method=>POST,
                           -action=>'reportoutput.cgi'),
                    hidden('ALoginID',$indata{'ALoginID'}),
                    hidden('Role',$indata{'Role'}),
                    hidden('Password',$indata{'Password'}),
                    hidden('DBMFILE',"./TICKETINDEX"),
                    hidden('report',$indata{'report'}),
                    br,
                    "Enter a Trouble Ticket #",br,
                    textfield(-name=>'Ticket',
                            -size=>11),
                    br,br,
                    submit(-name=>'Submit',
                           -value=>'Produce Report'),
                    br,
                end_form);
} elsif ($indata{'report'} eq 'Commonly Reported Problems') {
        print center(i(font({-size=>5}, $indata{'report'}, " Report")),
                start_form(-method=>POST,
                           -action=>'reportoutput.cgi'),
                    hidden('ALoginID',$indata{'ALoginID'}),
                    hidden('Role',$indata{'Role'}),
                    hidden('Password',$indata{'Password'}),
```

```
                        hidden('DBMFILE',"./TICKETINDEX"),
                        hidden('report',$indata{'report'}),
                        br,
                        "<TABLE>",
                        "<TR><TD>Start Date</TD><TD>",
                        textfield(-name=>'startdate',
                                  -value=>'MM-DD-YYYY',
                                  -size=>11),"</TD></TR>",
                        "<TR><TD>End Date</TD><TD>",
                        textfield(-name=>'enddate',
                                  -value=>'MM-DD-YYYY',
                                  -size=>11),"</TD></TR>",
                        "</TABLE>",br,
                        submit(-name=>'Submit',
                               -value=>'Produce Report'),
                        br,
                end_form);
} elsif ($indata{'report'} eq 'Keyword Search') {
        print center(i(font({-size=>5}, $indata{'report'}, " Report")),
                start_form(-method=>POST,
                           -action=>'reportoutput.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Role',$indata{'Role'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('DBMFILE',"./TICKETINDEX"),
                        hidden('report',$indata{'report'}),
                        br,
                        "<TABLE>",
                        "<TR><TD>Start Date</TD><TD>",
                        textfield(-name=>'startdate',
                                  -value=>'MM-DD-YYYY',
                                  -size=>11),"</TD></TR>",
                        "<TR><TD>End Date</TD><TD>",
                        textfield(-name=>'enddate',
                                  -value=>'MM-DD-YYYY',
                                  -size=>11),"</TD></TR>",
                        "<TR><TD>Keywords:</TD><TD>",
                        textarea(-name=>'Keywords',
                                  -rows=>4,
                                  -columns=>30,
                                  -wrap=>'physical'),"</TD></TR>",
                        "</TABLE>",br,
                        submit(-name=>'Submit',
                               -value=>'Produce Report'),
                        br,
                end_form);
}

print center(
        start_form(-action=>'adminmain.cgi'),
        hidden('ALoginID',$indata{'ALoginID'}),
        hidden('Role',$indata{'Role'}),
        hidden('Password',$indata{'Password'}),
        submit(-value=>'Return to the Main Menu'),
        end_form);

print end_html;
```

reportoutput.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to output the requested report for viewing by the Admin/Tech
#       who requested it.  If a hard copy is needed, the Admin/Tech may print it using
#       the browser's print capabilities.
#
#*****************************************************************************
use CGI qw/:standard :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System'),
                hr);

#       Read the Requested dbm (either OPEN or TICKETINDEX) to collect tickets within
the
#       requested date range
dbmopen(%ticketdata, $indata{'DBMFILE'}, 0644) || die "Can't access
",$indata{'DBMFILE'},", $!\n";
flock($indata{'DBMFILE'},2);
%reportdata = datecheck();
dbmclose(%ticketdata);
flock($indata{'DBMFILE'},8);

if ($indata{'report'} eq 'Full History') {
        print center(i(font({-size=>5}, "Full History Report from
",$indata{'startdate'},"  to ",$indata{'enddate'}))),br;
        @reporttkts = keys(%reportdata);
        report1();
} elsif ($indata{'report'} eq 'Open Trouble Tickets') {
        print center(i(font({-size=>5}, "Open Trouble Tickets Assigned to
",$indata{'AssignedTo'}))),br;
        foreach (keys(%reportdata)) {
                if (($reportdata{$_} eq $indata{'AssignedTo'}) ||
($indata{'AssignedTo'} eq "All")) {
                        push (@reporttkts, $_);
                }
        }
        report1();
} elsif ($indata{'report'} eq 'Tickets Submitted by a Specific User') {
        print center(i(font({-size=>5}, "Trouble Tickets Submitted By
",$indata{'UserName'},"  from ",$indata{'startdate'},"  to ",$indata{'enddate'}))),br;
        foreach (keys(%reportdata)) {
                if ($_ =~ /$indata{'UserName'}/) {
                        push (@reporttkts, $_);
                }
        }
        report1();
} elsif ($indata{'report'} eq 'Trouble Ticket Audit Trail') {
        print center(i(font({-size=>5}, "Audit Trail for Ticket#
",$indata{'Ticket'}))),br;
        $garbage = currentstatus($indata{'Ticket'});
        report2();
} elsif ($indata{'report'} eq 'Commonly Reported Problems') {
        print center(i(font({-size=>5}, "Commonly Reported Problems from
",$indata{'startdate'},"  to ",$indata{'enddate'}))),br;
        report3();
} elsif ($indata{'report'} eq 'Keyword Search') {
```

```
        print center(i(font({-size=>5}, "Keyword Search for ",$indata{'Keywords'},"
from ",$indata{'startdate'}," to ",$indata{'enddate'}))),br;
        @keywords = split (/ /, $indata{'Keywords'});
        report4();
}

print center(start_form(-method=>POST,
                    -action=>'adminmain.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                br,
                submit(-name=>'Return to the Main Menu'),
        end_form);

sub datecheck {
#*****************************************************************************************
#       This subroutine is used to find the Trouble Tickets that fall within the
#       requested date range.
#*****************************************************************************************
local %reportkeys;

        ($smon,$sday,$syear) = split (/-/, $indata{'startdate'});
        ($emon,$eday,$eyear) = split (/-/, $indata{'enddate'});
        foreach (keys(%ticketdata)) {
                @tempdate = split (/-/, $_);
                $montemp = strftime('%m',localtime(@tempdate[1]));
                $daytemp = strftime('%d',localtime(@tempdate[1]));
                $yeartemp = strftime('%Y',localtime(@tempdate[1]));
                if ((($syear <= $yeartemp) && ($yeartemp <= $eyear)) && (($smon <=
$montemp) && ($montemp <= $emon)) && (($sday <= $daytemp) && ($daytemp <= $eday))) {
                        $reportkeys{$_} = $ticketdata{$_};
                }
        }
        return %reportkeys;
}

sub report1 {
#*****************************************************************************************
#       This is the report format used for the Full History, Open Tickets, and Tickets
#       submitted by a specific User Reports.
#*****************************************************************************************
        local $key;
        local @keytemp,@valuetemp;

        foreach (sort @reporttkts) {
                @keytemp = split (/~/, $_);
                $lastticket = currentstatus(@keytemp[0]);
                @valuetemp = split (/~/, $lastticket);

#       This if-elsif statement is used to set the hash key to the proper sorting
field
                if ($indata{'Sort'} eq "User") {
                        $key =
sprintf("%s~%s",@keytemp[2],(sprintf("%08d",@keytemp[0])));
                } elsif ($indata{'Sort'} eq "Ticket#") {
                        $key = (sprintf("%08d",@keytemp[0]));
                } elsif ($indata{'Sort'} eq "Priority") {
                        $key =
sprintf("%s~%s",@valuetemp[7],(sprintf("%08d",@keytemp[0])));
                }
                $report{$key} =
sprintf("%s~%s~%s~%s~%s~%s~%s",@keytemp[0],strftime('%c',localtime(@keytemp[1])),@keyt
emp[2],@valuetemp[7],@valuetemp[0],@valuetemp[5],@valuetemp[8]),br;
        }

#       Display the proper table heading
        print center("<TABLE border=1>",
        "<TH WIDTH=7% ALIGN=LEFT><B>Ticket#</B></TH><TH WIDTH=25% ALIGN=LEFT><B>Date &
Time Submitted</B></TH><TH WIDTH=10% ALIGN=LEFT><B>User</B></TH><TH WIDTH=10%
ALIGN=LEFT><B>Priority</B></TH><TH WIDTH=28% ALIGN=LEFT><B>Problem
```

```
Description</B></TH><TH WIDTH=10% ALIGN=LEFT><B>Assigned To</B></TH><TH WIDTH=10%
ALIGN=LEFT><B>Status</B></TH></TR>");
        printreport();
}

sub report2 {
#*******************************************************************************
#       This is the report format used for the Trouble Ticket Audit Trail Report.
#*******************************************************************************
        local $key;
        local @keytemp,@valuetemp;

        dbmopen(%ticketdata, "./TICKETS", 0644) || die "Can't access TICKETS, $!\n";
        flock(TICKETS,2);
        foreach (sort @tkttemp) {
#       Collect all records for the requested Trouble Ticket.
                @keytemp = split (/~/, $_);
                @valuetemp = split (/~/, $ticketdata{$_});
                $report{$_} =
sprintf("%s~%s~%s~%s~%s",strftime('%c',localtime(@keytemp[1])),@valuetemp[7],@valuetem
p[5],@valuetemp[8],@valuetemp[6]);
        }
        dbmclose(%ticketdata);
        flock(TICKETS,8);

#       Display the proper table heading
        print center("<TABLE border=1>",
        "<TH WIDTH=25% ALIGN=LEFT><B>Date & Time Submitted</B></TH><TH WIDTH=10%
ALIGN=LEFT><B>Priority</B></TH><TH WIDTH=10% ALIGN=LEFT><B>Assigned To</B></TH><TH
WIDTH=10% ALIGN=LEFT><B>Status</B></TH><TH WIDTH=45%
ALIGN=LEFT><B>Comments</B></TH></TR>");
        printreport();
}

sub report3 {
#*******************************************************************************
#       This is the report format used for the Commonly Reported Problems Report.
#*******************************************************************************
        local $count,$total, $errcount;


        $total = scalar(keys(%reportdata));
        dbmopen(%cattemp, "./CATEGORIES", 0644) || die "Can't access CATEGORIES,
$!\n";
        flock(CATEGORIES,2);
        foreach $cat (keys(%cattemp)) {
#       Loop through each Category
                $count = 0;
                foreach (values(%reportdata)) {
#       Loop through each selected record to check for a Category match
                        if ($cat eq $_) {
                                $count += 1;
                        }
                }
                $report{$cat} =
sprintf("%s~%s~%3.2f%",$cat,$count,(($count/$total)*100));
        }
        dbmclose(%cattemp);
        flock(CATEGORIES,8);

#       Display the proper table heading
        print center("<TABLE border=1>",
        "<TH WIDTH=20% ALIGN=LEFT><B>Category</B></TH><TH WIDTH=10% ALIGN=LEFT><B># of
Tickets</B></TH><TH WIDTH=10% ALIGN=LEFT><B>% of Tickets</B></TH></TR>");
        printreport();
}

sub report4 {
#*******************************************************************************
#       This is the report format used for the Keyword Search Report.
#*******************************************************************************
```

```
        local @keytemp,@valuetemp;

        foreach (keys %reportdata) {
#       Loop used to get all records matching all Trouble Ticket numbers in the
requested date range
                currentstatus($_ =~ m/^([\d]+)/);
        }
        dbmopen(%ticketdata, "./TICKETS", 0644) || die "Can't access TICKETS, $!\n";
        flock(TICKETS,2);
        foreach $tkt (@tkttemp) {
#       Loop to check each record for each of the requested Keywords
                foreach $kw (@keywords) {
                        if ($ticketdata{$tkt} =~ /$kw/) {
                                @keytemp = split (/~/, $tkt);
                                $keywordrecs{@keytemp[0]} = $tkt;
                        }
                }
        }
        dbmclose(%ticketdata);
        flock(TICKETS,8);
        foreach (keys %keywordrecs) {
#       Loop to collect the current data for each of the records containing a Keyword
                $lastticket = currentstatus($_);
                @keytemp = split (/~/, $keywordrecs{$_});
                @valuetemp = split (/~/, $lastticket);
                $report{(sprintf("%08d",$_))} =
sprintf("%s~%s",@keytemp[0],@valuetemp[0]);
        }

#       Display the proper table heading
        print center("<TABLE border=1>",
        "<TH WIDTH=7% ALIGN=LEFT><B>Ticket#</B></TH><TH WIDTH=28%
ALIGN=LEFT><B>Problem Description</B></TH></TR>");
        printreport();
}

sub currentstatus {
#*******************************************************************************
#       This subroutine is used to find the current status of a Trouble Ticket by
#       reviewing the data in the last record for that ticket.
#*******************************************************************************
local @tktholder;

        dbmopen(%ticketdata, "./TICKETS", 0644) || die "Can't access TICKETS, $!\n";
        flock(TICKETS,2);
        for (keys(%ticketdata)) {
#       Loop to collect each record for the requested Trouble Ticket number
                if ((sprintf("%s",$_ =~ m/^([\d]+)/)) eq $_[0]) {
                        push (@tktholder, $_);
                        push (@tkttemp, $_);
                }
        }
        @tktholder = sort @tktholder;
        @tkttemp = sort @tkttemp;

#       Determining the last record for the Trouble Ticket
        $lasttkt = $ticketdata{@tktholder[(scalar(@tktholder)-1)]};
        dbmclose(%ticketdata);
        flock(TICKETS,8);
        return $lasttkt;
}

print end_html;
```

reports.cgi

```
#!/usr/bin/perl
#*******************************************************************************
#
#       Tom Braden     April 2000
#
#       This page is used to select the Report to be generated.
#
#*******************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

print center(i(font({-size=>5}, "Please select a report below.")),
        start_form(-method=>POST,
                   -action=>'reportdata.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                br,
                popup_menu(-name=>'report',
                           -value=>[' ','Full History','Open Trouble Tickets','Trouble
Ticket Audit Trail','Tickets submitted by a specific User','Commonly Reported
Problems','Keyword Search'],
                           -default=>' '),
                br,br,
                submit(-name=>'Submit',
                       -value=>'Select Report'),
        end_form);

print center(start_form(-method=>POST,
                   -action=>'adminmain.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                submit(-name=>'Return to the Main Menu'),
        end_form);

print end_html;
```

ticketselectstatus.cgi

```
#!/usr/bin/perl
#*********************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to request the Trouble Ticket number to be reviewed.
#
#*********************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
        hr);

print center(i(font({-size=>5}, "Please enter the Trouble Ticket # that you would like
to review.")),
        start_form(-method=>POST,
                   -action=>'ticketstatus.cgi'),
                hidden('LoginID',$indata{'LoginID'}),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                br,
                textfield(-name=>'Ticket',
                          -size=>11),
                br,br,
                submit(-name=>'Submit',
                       -value=>'Check the Status of this Ticket'),
                br,
        end_form);

if ($indata{'LoginID'} ne "") {
#       Provide a link back to the User Main Menu if the page is being used by a User
        print center(
                start_form(-method=>POST,
                           -action=>'usermain.cgi'),
                hidden('LoginID',$indata{'LoginID'}),
                submit(-value=>'Return to the Main Menu'),
                end_form);

} elsif ($indata{'ALoginID'} ne "") {
#       Provide a link back to the Admin/Tech Main Menu if the page is being used by a
Admin/Tech
        print center(
                start_form(-method=>POST,
                           -action=>'adminmain.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                submit(-value=>'Return to the Main Menu'),
                end_form);
}


print end_html;
```

ticketstatus.cgi

```
#!/usr/bin/perl
#*****************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to display each record associated with a Trouble Ticket.
#       The requester can then view the status of the ticket each time it was modified
#       to see what changes were made and by who, as well as the current status.
#
#*****************************************************************************
use CGI qw/:standard :netscape/;
use POSIX 'strftime';
require 'idts.lib';

%indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

#       Collect all records for the requested Trouble Ticket
$ticketnum = sprintf("%s",$indata{'Ticket'} =~ m/^([\d]+)/);
dbmopen(%ticketdata, "./TICKETS", 0644) || die "Can't access TICKETS, $!\n";
flock(TICKETS,2);
for (keys(%ticketdata)) {
        if ((sprintf("%s",$_ =~ m/^([\d]+)/)) eq $ticketnum) {
                @keytemp = split (/~/, $_);
                @valuetemp = split (/~/, $ticketdata{$_});
                $report{$_} =
sprintf("%s~%s~%s~%s~%s~%s~%s",strftime('%c',localtime(@keytemp[1])),@keytemp[2],@valuete
mp[4],@valuetemp[7],@valuetemp[5],@valuetemp[8]);
                push (@tktstatus, $_);
        }
}
dbmopen(%opendata, "./OPEN", 0644) || die "Can't access OPEN, $!\n";
flock(OPEN,2);

#       Collect user data for each ID associated with the requested ticket
foreach (keys(%opendata)) {
        if ((sprintf("%s",$_ =~ m/^([\d]+)/)) eq $ticketnum) {
                @user = split (/~/, $_);
        }
}
dbmclose(%opendata);
flock(OPEN,8);
dbmopen(%userdata, "./USERS", 0644) || die "Can't access USERS, $!\n";
flock(USERS,2);
        @usertemp = split (/~/, $userdata{@user[2]});
dbmclose(%userdata);
flock(USERS,8);

print center(br,i(font({-size=>5}, "Ticket Number:",sprintf("%s,
%s",$ticketnum,$ticketdata{@tktstatus[0]} =~ m/^([^~]+)/))),
        start_form,br);
        if (@usertemp[0] ne "") {
                print center("<TABLE border=1>",
                "<TR><TD>Name</TD><TD>",@usertemp[0]," ",@usertemp[1],"</TD>",
                "<TR><TD>Phone</TD><TD>",@valuetemp[1],"</TD>",
                "<TR><TD>Building</TD><TD>",@valuetemp[2],"</TD>",
                "<TR><TD>Room</TD><TD>",@valuetemp[3],"</TD>",
                "</TABLE>",br);
        }
        print center ("<TABLE border=1>",
```

```
        "<TH WIDTH=25% ALIGN=LEFT><B>Date & Time Submitted</B></TH><TH WIDTH=10%
ALIGN=LEFT><B>Changed By</B></TH><TH WIDTH=20% ALIGN=LEFT><B>Category</B></TH><TH
WIDTH=10% ALIGN=LEFT><B>Priority</B></TH><TH WIDTH=10% ALIGN=LEFT><B>Assigned
To</B></TH><TH WIDTH=10% ALIGN=LEFT><B>Status</B></TH></TR>");
        printreport();

dbmclose(%ticketdata);
flock(TICKETS,8);

if ($indata{'LoginID'} ne "") {
        print center(start_form(-method=>POST,
                        -action=>'usermain.cgi'),
                hidden('LoginID',$indata{'LoginID'}),
                br,
                submit(-name=>'Return to the Main Menu'),
        end_form);
} elsif ($indata{'ALoginID'} ne "") {
        print center(start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                hidden('ALoginID',$indata{'ALoginID'}),
                hidden('Password',$indata{'Password'}),
                hidden('Role',$indata{'Role'}),
                br,
                submit(-name=>'Return to the Main Menu'),
        end_form);
}


print end_html;
```

ticketsubmit.cgi

```
#!/usr/bin/perl
#*****************************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to collect information to create a Trouble Ticket.  The user
#       enters the information in this form while User data is recorded from the USERS
#       dbm.
#
#*****************************************************************************************
use CGI qw/:standard :html2 :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
        hr);

#       Read the CATEGORIES dbm to dynamically create a popup menu
dbmopen(%categorydata, "./CATEGORIES", 0644) || die "Can't access CATEGORIES, $!\n";
flock(CATEGORIES,2);
%temphash = %categorydata;
dbmclose(%categorydata);
flock(CATEGORIES,8);
#       Read the USERS dbm to retrieve data about the user submitting the ticket
dbmopen(%userdata, "./USERS", undef) || die "Can't access USERS, $!\n";
flock(USERS,2);
@usertemp = split (/~/, $userdata{$indata{'LoginID'}});
dbmclose(%userdata);
flock(USERS,8);

print center(i(font({-size=>5}, "Please complete and submit the form below.")),
        start_form(-method=>POST,
                   -action=>'ticketsubmitconf.cgi'),
                hidden('LoginID',$indata{'LoginID'}),
                hidden('Status',"Open"),
                hidden('AssignedTo',"TBD"),
                hidden('tktnum',"00"),
                br,
                "<TABLE>\n",
                "<TR><TD>Name:</TD><TD>",$indata{'LoginID'},"</TD></TR>",
                "<TR><TD>Phone:</TD><TD>",
                textfield(-name=>'UserPhone',
                          -value=>@usertemp[2],
                          -size=>30),"</TD></TR>",
                "<TR><TD>Building:</TD><TD>",
                textfield(-name=>'UserBldg',
                          -value=>@usertemp[3],
                          -size=>30),"</TD></TR>",
                "<TR><TD>Room:</TD><TD>",
                textfield(-name=>'UserRoom',
                          -value=>@usertemp[4],
                          -size=>30),"</TD></TR>",
                "<TR><TD>Category:</TD><TD>",
                popup_menu(-name=>'TroubleCategory',
                          -value=>['Not Sure',sort keys(%temphash)],
                          -default=>'Not Sure'),"</TD></TR>",
                "<TR><TD>Problem:</TD><TD>",
                textfield(-name=>'ProbBrief',
                          -size=>30),"</TD></TR>",
                "<TR><TD>Problem Description:</TD><TD>",
                textarea(-name=>'TroubleDescription',
```

```
                        -rows=>5,
                        -columns=>40,
                        -wrap=>'physical'),"</TD></TR>",
                "<TR><TD>Priority:</TD><TD>",
                popup_menu(-name=>'TroublePriority',
                        -value=>['1 - Critical','2 - High','3 - Moderate','4 -
Low','5 - Very Low'],
                        -default=>'3 - Moderate'),"</TD></TR>",
                "</TABLE>",
                br,
                submit(-name=>'Submit',
                       -value=>'Submit this Ticket'),br,br,
                reset(-name=>'Clear this Ticket'),
                br,
        end_form);

print center(
        start_form(-action=>'usermain.cgi'),
        hidden('LoginID',$indata{'LoginID'}),
        submit(-value=>'Return to the Main Menu'),
        end_form);

print end_html;
```

ticketsubmitconf.cgi

```
#!/usr/bin/perl
#*****************************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to create or update a Trouble Ticket.  This requires
#       updating the information about the Trouble in the TICKETS, TICKETINDEX and
#       OPEN dbms.
#
#*****************************************************************************************
use CGI qw/:standard :netscape/;
use POSIX 'strftime';
require 'idts.lib';

%indata = getcgivars();

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br,$indata{'Role'}),
                hr);

dbmopen(%ticketdata, "./TICKETS", 0644) || die "Can't access TICKETS, $!\n";
flock(TICKETS,2);
dbmopen(%ticketindex, "./TICKETINDEX", 0644) || die "Can't access TICKETINDEX, $!\n";
flock(TICKETINDEX,2);
dbmopen(%openticket, "./OPEN", 0644) || die "Can't access OPEN, $!\n";
flock(OPEN,2);

if ($indata{'tktnum'} eq "00") {
#       If this is a new ticket, a new ticket number must be generated
        $tktnum = scalar(keys(%ticketindex)) + 1;
} else {
        $tktnum = $indata{'tktnum'};
}
$tkttime = time;
$key = sprintf("%s~%d~%s",$tktnum,$tkttime,$indata{'LoginID'});
if (($indata{'tktnum'} eq "00") || ($indata{'ReDispatch'} eq "Yes")) {
#       If this is a new ticket or re-dispatch request, it must be dispatched to an
Admin/Tech
        $indata{'AssignedTo'} = dispatch();
}
if ($indata{'tktnum'} eq "00") {
#       If this is a new ticket, the Assigned Tech and Category must be saved
        $openticket{$key} = $indata{'AssignedTo'};
        $ticketindex{$key} = $indata{'TroubleCategory'};
} else {
#       If this is an existing ticket, the Assigned Tech and Category must be saved in
case
#       they were changed
        for (keys(%openticket)) {
                if ($indata{'Ticket'} eq sprintf("%s",$_ =~ m/^([^~]+)/)) {
                        $openticket{$_} = $indata{'AssignedTo'};
                        $ticketindex{$_} = $indata{'TroubleCategory'};
                        if ($indata{'Status'} =~ /Closed/) {
                                delete($openticket{$_});
                        }
                }
        }
}

dbmclose(%openticket);
flock(OPEN,8);
dbmclose(%ticketindex);
flock(TICKETINDEX,8);
```

```
$value =
sprintf("%s~%s~%s~%s~%s~%s~%s~%s~%s~%s",$indata{'ProbBrief'},$indata{'UserPhone'},$indata
{'UserBldg'},$indata{'UserRoom'},$indata{'TroubleCategory'},$indata{'AssignedTo'},$ind
ata{'TroubleDescription'},$indata{'TroublePriority'},$indata{'Status'});
$ticketdata{$key} = $value;
dbmclose(%ticketdata);
flock(TICKETS,8);
if ($indata{'tktnum'} eq "00") {
        print center(br,i(font({-size=>5}, "Ticket Number ",$tktnum," was submitted
at",strftime('%I:%M %p on %A, %B %d %Y',localtime($tkttime)),".")));
} else {
        print center(br,i(font({-size=>5}, "Ticket Number ",$tktnum," was updated
at",strftime('%I:%M %p on %A, %B %d %Y',localtime($tkttime)),".")));
}

if ($indata{'ALoginID'} eq "") {
#       If this page was called by a user, return to the User's Main Menu
                print center(start_form(-method=>POST,
                        -action=>'usermain.cgi'),
                        hidden('LoginID',$indata{'LoginID'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
} else {
#       If this page was called by an Admin/Tech, return to the Admin's/Tech's Main
Menu
                print center(start_form(-method=>POST,
                        -action=>'adminmain.cgi'),
                        hidden('ALoginID',$indata{'ALoginID'}),
                        hidden('Password',$indata{'Password'}),
                        hidden('Role',$indata{'Role'}),
                        br,
                        submit(-name=>'Return to the Main Menu'),
                end_form);
}

sub dispatch {
#*****************************************************************************
#       This subroutine is used to dispatch a new trouble ticket to an Admin/Tech.
#       The process determines all of the Admin/Techs who handle troubles in the
#       tickets trouble category and assigns it to the one with the fewest number of
#       open tickets.
#*****************************************************************************
        $tktquantity = 1000000;
        dbmopen(%specs, "./ADMINSPEC", 0644) || die "Can't access ADMINSPEC, $!\n";
        flock(ADMINSPEC,2);
        for (sort %specs) {
                if ($specs{$_} =~ /$indata{'TroubleCategory'}/) {
                        $tech = $_;
                        $techopen = 0;
                        for (%openticket) {
                                if ($openticket{$_} eq $tech) {
                                        $techopen += 1;
                                }
                        }
                        if ($techopen < $tktquantity) {
                                $assigntech = $tech;
                                $tktquantity = $techopen;
                        }
                }
        }
        if ($tktquantity eq 1000000) {
#       The ticket cannot be dispatched and will need to be assigned by an
Administrator
                $assigntech = "TBD";
        }
        dbmclose(%specs);
        flock(ADMINSPEC,8);
        return $assigntech;
}
print end_html;
```

userlogin.cgi

```
#!/usr/bin/perl
#*********************************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to collect the User's ID for login.
#
#*********************************************************************************************
use CGI qw/:standard :netscape/;

print header,

        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'fffccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br),
                hr);

print center(i(font({-size=>5}, "Please enter your ID to Log on to the system.")),
        start_form(-method=>POST,
                    -action=>'usermain.cgi'),
                "<TABLE>",
                "<TR><TD>Username:</TD><TD>",
                textfield('LoginID'),"</TD></TR>",
                "</TABLE>",
                br,
                submit(-name=>'Submit',
                        -value=>'LogOn'),
                br,
        end_form);

print end_html;
```

usermain.cgi

```
#!/usr/bin/perl
#*********************************************************************************************
#
#       Tom Braden      April 2000
#
#       This page is used to display the main menu for a User.
#
#*********************************************************************************************
use CGI qw/:standard :html3 :netscape/;
use POSIX 'strftime';
require 'idts.lib';

my %indata = getcgivars();

print header,
        start_html(-title=>'The Incident Dispatching and Tracking System',
                -BGCOLOR=>'ffcccc',
                -TEXT=>'blue'),
        center(h1('The Incident Dispatching and Tracking System',br),
        hr);
#       Tying the USERS dbm to a hash for use in identifying the user
dbmopen(%userdata, "./USERS", undef) || die "Can't access USERS, $!\n";
flock(USERS,2);
%temphash = %userdata;
@usertemp = split (/~/, $userdata{$indata{'LoginID'}});
dbmclose(%userdata);
flock(USERS,8);
if (exists ($temphash{$indata{'LoginID'}})) {
#       Display the User's Main Menu
        print center(
                start_form,
                "Welcome, ",@usertemp[0]," ",@usertemp[1],
                br,
                font({-size=>5}, "Please make your selection below"),
                br,
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'ticketsubmit.cgi'),
                hidden('LoginID',$indata{'LoginID'}),
                submit(-value=>'Submit a Trouble Ticket'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'lookuptickets.cgi'),
                hidden('LoginID',$indata{'LoginID'}),
                submit(-value=>'Lookup a Trouble Ticket'),
                end_form);
        print center(
                start_form(-method=>POST,
                        -action=>'ticketselectstatus.cgi'),
                hidden('LoginID',$indata{'LoginID'}),
                submit(-value=>'Check a Trouble Ticket'),
                end_form);
        print center(
                start_form(-action=>'userlogin.cgi'),
                submit(-value=>'Logout'),
                end_form);
} else {
#       Error - the username does not exist
        print center(
                start_form(-action=>'userlogin.cgi'),
                "Username: ",$indata{'LoginID'}," does not exist",
                br,
                submit(-value=>'Return to User Login'),
                end_form);
}
print end_html;
```

Appendix B - DBM Files

- ADMINS - The list of Admin/Techs.
- ADMINSPEC - An index used to easily identify the specialties of each Admin/Tech.
- CATEGORIES - The list of categories used to define the type of trouble ticket being submitted and to define the specialties of each Admin/Tech.
- OPEN - All open trouble tickets. The key matches the TICKETS key generated when a trouble ticket is first submitted matched to the ID of the Admin/Tech that the ticket is assigned to.
- TICKETS - All ticket information and history. The key contains the ticket number, timestamp and ID for each record generated by a new ticket or ticket update matched to the collected ticket information.
- TICKETINDEX - A list of the original TICKETS key generated when a new ticket matched to the category that the ticket is defined as.
- USERS - The contact and location information of each User.