Helpdesk.Drew.Edu: Home Growing a Helpdesk Solution Using Open-Source Technology

John Saul
Drew University
36 Madison Ave.
Madison, NJ 079401
-973-408-3146
jsaul@drew.edu

Betsy Black
Drew University
36 Madison Ave.
Madison, NJ 079401
-973-408-3860
eblack@drew.edu

Erik Larsson
Drew University
36 Madison Ave.
Madison, NJ 079401
-973-408-5094
elarsson@drew.edu

ABSTRACT

In July of 1999, the department of Academic Technology at Drew University began the search for a trouble-ticket tracking package. After researching several commercially available products, we determined that none offered us the flexibility we needed at a price we could afford. Despite our lack of success in finding a commercially available product, our need for an effective trouble-ticket package was becoming more pronounced.

In August of 1999 the Helpdesk project was created to write our own web-based trouble-ticket tracking package. The project originated from a pair of students, Erik Larsson (CLA AA02) and Jessica Sockel (CLA '00), who suggested we use open-source technology to produce our own web-based tracking system.

We obtained space on a web server and named our project helpdesk.drew.edu. At this point the Helpdesk project became modified to include asset tracking and management, as well as a discussion forum.

We chose PHP as the primary coding language. Our choice of MySQL as a database engine was based on our need to have a fast, reliable database with which PHP could easily interface. For the discussion forum we used and modified an open-source product called Phorum which uses PHP and MySQL as well.

At present, helpdesk.drew.edu has been in use successfully for two semesters. We plan to refine and improve how problems are tracked through this system. Our processes and solutions can be modified and adapted by the helpdesk of any school that doesn't have a large budget to spend for a trouble-ticket tracking package or is unhappy with the adaptability of their current trouble-ticket software. Our paper is intended to further explain the process that we went through in choosing the specific open-source solutions, as well as the problems and benefits we encountered during the development process.

LEAVE THIS TEXT BOX IN PLACE
AND BLANK

Keywords

Helpdesk, open-source technology, trouble tracking, tickets, PHP, MySQL

1. INTRODUCTION

Drew University began providing its students with computers in 1984 as part of its Computer Initiative (CI) Program. These computers are distributed as standard packages and are supported for the student●s full term at Drew. The Aide Station desk, staffed entirely by students, is the University's primary helpdesk operation and supports only the software and hardware that are given as part of each computer's configuration.

The Computer Aide Station has been supporting the computer-related needs of the University since the inception of the CI. The Aide Station supports about 2,000 students and about 500 faculty and staff members.

To facilitate communication and provide a central location for a knowledge base, the support staff began using an OpenVMS DEC NOTES system in 1992. Aide Station operators (ATOPS) would post information to this internal system about various issues dealing with the Computer Center, as well as solutions to problems encountered on a day-to-day basis. Other operators could then use single word or phrase searching to pull up that information for future reference, or reply to questions posted by others.

This system worked admirably well for years until the increased complexity of operating systems, software packages, and the technical ability of end users combined with the staged implementation of a Novell Netware local area network began causing delays in the resolution of technical problems. The NOTES system was beginning to show its age; problems began to go unresolved for far too long; and computers that were left with us for repair--or what we termed "intaken"--would sometimes get misplaced or parts would get lost.

In 1998, with the completion of the Residential Network and the addition of a network interface card in each CI package, the number of problems and intaken computers began to rise to levels never before seen. At the beginning of the fall semester of 1998, more than 30 computers were intaken in one day by a desk manned by two ATOPS. After the initial rush of computers, things began to slow down, but there was still a noticeable increase in the amount of computers being intaken. The need for an automated system became

apparent as early as 1997; a search for software began, but the project was postponed due to lack of funds and personnel to learn and maintain the system.

2. THE SEARCH PROCESS

2.1 Commercially Available Packages

The search was revived in June of 1999 when Academic Technology created the PC Support Services division, promoted Betsy Black to manager, and hired John Saul as Software Support Specialist. Saul was charged with reviewing the various commercially available packages and contacting vendors to determine what features each offered. Saul was also asked to help identify features that would be required for the new system, including inventory tracking, trouble ticket tracking, and a discussion area for full-time staff and ATOPS.

Saul reduced the list to three commercially available products: HEAT◀◀ by the Goldmine Corporation, Remedy◀◀ by the Remedy Corporation, and HelpSTAR◀◀ by HelpSTAR.com. Each product offered at least three-quarters of the feature set we required, including incident tracking, asset management, and affordablility.

Demo models of each package were acquired and tested by members of both the full-time and student staff of the Computer Aide Station. Recommendations were compiled and a finalist was selected. The process to obtain specific pricing and resource requirements for the package was then begun.

Unfortunately, the final cost to the department was found to be beyond our means. Efforts were made to manage the cost, since the need for an incident tracking package was quickly becoming more pronounced, but the licensing, hardware requirements, and maintenance/upgrade costs were ballooning far outside of our budget.

2.2 Developing Our Own

Since we had already identified the components necessary for the ideal program, Aide Station student managers Erik Larsson (CLA '02) and Jessica Sockel (CLA '00) proposed writing our own web-based tracking system after discovering Phorum, a web-based discussion software written in PHP▶▶. Unlike most popular web boards, Phorum utilizes a database to manage its messages. This allows easy administration, functionality and customization. The most attractive feature was its price: Phorum is an open-source program necessitating only agreement to their licensing requirements.

Phorum uses either MySQL or PostGRESQL to store the message information. A copy of MySQL was already installed and working on one of the systems in the department, so MySQL was the easiest database engine to use for development.

3. DESIGNING THE SYSTEM

3.1 Required Feature Set

For the helpdesk software to be useful, the desired features should be clearly defined before any designing begins. We defined three major features necessary for our software: the ability to track intaken computers via tickets along with their components (network card and dongle, external floppy drive, etc.); record any actions taken to solve problems; and keep track of where in the repair process the computer is. All tickets needed to be tracked by the user that initiated them so that recurring problems -- either software- or user-related -- could be easily tracked.

Additionally, we decided that it would be useful for us to track all problems attached to a specific computer model, regardless of which user intiated the ticket. Since the method of upgrading computers for faculty and staff works in a 🎬roll-down🛠 fashion, one computer often has multiple owners over its lifetime at the University. By 🎬roll-down🛠 we mean that the user getting a brand new computer gives up their current machine, which gets reassigned to a user with an older computer and so on down the line.

To track the assets effectively, we needed to store the customer and departmental information for each asset. Any piece of equipment could then be tracked based on the department to which it belonged, and then anyone within that department could initiate a ticket on any of those assets.

3.2 The Initial Stages and Unique Issues with Development

Near the end of July, a new virtual host, helpdesk.drew.edu, was christened on one of the department's Linux-based web servers. Work on the project progressed swiftly and was completed by the end of August in time for ATOPS training. The software was uniquely customized to the Drew environment in several respects.

3.2.1 Authentication and user administration.

Drew●s campus network is filled with disparate systems, running various operating systems and supporting assorted campus services. At the time of Helpdesk●s inception, campus network users had to remember at least two passwords: A Novell NetWare password, and a separate password for our UNIX-based e-mail server. Some users, including ATOPS and Academic Technology staff, also had separate passwords for other OpenVMS and UNIX systems used for various purposes within the department. Rather than require users to remember 🎬Yet Another Password,🛠 it was imperative that the Helpdesk system authenticate with a user●s Novell Directory Services (NDS) password. This goal was accomplished using Novell●s Lightweight Directory Access Protocol (LDAP) server for NDS with PHP●s integrated LDAP support.

Authentication requests are made directly against Drew●s NDS database, with no need for separate passwords or any password synchronization. A further requirement was that passwords never be transmitted over the network in plain text. Conversely, the web server on which we were developing the software would have sustained too much of a performance hit if all transactions with Helpdesk were conducted over an encrypted Secure Socket Layer (SSL) connection. Our solution was to process logins through a web form rather than using HTTP Basic Authentication and to then use SSL only for that form.

Once a user●s password is verified with NDS, a randomly generated session key is created and returned to the browser as a cookie while also being stored in a session control table in the MySQL database. Further requests to the Helpdesk system involve only checking the session key, thereby avoiding additional NDS queries. Session keys are destroyed either by the user explicitly clicking a logout button, or after one hour of inactivity. Subsequent visits to Helpdesk will prompt for re-authentication, and a new session key will be assigned.

A further requirement of the system was that it support varying levels of privilege for different users, such as administrators, ATOPS, and customers. This requirement was accomplished using a very simple model of explicitly assigned user privileges. The Helpdesk database includes a user table, which simply lists the privileges that are assigned to the user. Privileges are checked by the modules that require them, and a web-based administrative interface is provided for creating and deleting users and assigning privileges. The assignable privileges are:

- login - is used for all full-time and student staff who log into the Helpdesk
- tickets - allows a user to create, modify, and close a ticket
- edit_closed_tickets - allows a user to re-open or otherwise change the status on a closed ticket
- inventory - allows a user to modify the equipment tables
- admin - is used for administrators of the Helpdesk

3.2.2 Multiple user access to records.

Any number of users may be logged in and actively using the Helpdesk system during the day. Occasionally, although infrequently, it is possible that two or more users may open the same record for editing simultaneously. In order to avoid having users overwrite each other's changes, some form of a record locking system was required. Using a traditional record-locking approach would not work adequately in a web-based environment. Due to the stateless nature of the web, there is no way for the web server to "know" when a connection has been terminated. Users may freely browse to other web sites, their computer may crash, or they may simply close their browser without saving changes to an open record. If we had used traditional record locking technique with an arbitrary time limit, users could lose their lock if taking too long to edit a record, whereas others would be locked out of a record after a computer crash while a customer waited on the phone. Traditional record locking is also impractical in our situation because users will often want to leave a record open during the entire time they are working on a computer, yet will require near immediate release of a lock in the event of a crash.

Our solution, which we termed "record overwrite protection," was to store the state of a record when a user enters a form and compare that state of information with the current state of the record in the database before allowing changes to be committed. That is, if another user commits changes to a record while you are working on it, your changes will be disallowed. In this situation, users are presented with an error explaining what happened and are given the opportunity to manually merge their modifications with the current record.

3.3 Secondary Development Issues

3.3.1 Pre-populating fields in tickets

When editing a record, there are numerous status/checkbox fields that need to reflect the current state of the record. Normal static web forms would not have allowed us to maintain and display the dynamic nature our component data, since they only allow one state at any one time. The default state would have to be blank in all checkboxes to avoid the appearance that we lost a component.

Another suggestion was to disallow any changes to the components field of the initial form. This method was unfeasible , because there are times when additional components need to be brought to the Aide Station for testing. The ticket should reflect these additional components.

Fortunately, PHP has a way of allowing users to dynamically change the HTML code of a web page dependent upon the information that the script receives from the previous page. Using this option, we managed to have a form that automatically makes sure all checkboxes for components belonging to a ticket are checked. The status flag would be set to the correct value when an existing ticket was opened for editing.

This same feature allowed us to have the "steps taken" and "internal comments" fields available for editing by anyone working on the ticket. Since we were planning on allowing customer access to view the "steps taken" portion of the ticket, we needed to make sure that all comments made were professional in nature.

3.3.2 Customer access and notification

To minimize phone inquiries to the Aide Station desk, we decided to automate a system to keep our customers informed of every stage of their computer's repair. Using PHP scripts, the Helpdesk will email the customer each time the status of their ticket changes. The email includes all the comments in the "Steps Taken" field of the ticket as well as the current status; i.e., sent to manufacturer, waiting for parts, pending, etc.

Using their NDS username and password, customers can log into a Helpdesk customer access page and view all tickets, active and closed, that were opened under their username. The query returns a screen listing all departmental computers as well as a summary of every ticket that has been opened under that username.

3.3.3 Ticket receipts

In order to protect ourselves from erroneous claims of lost equipment, customers must sign a form indicating which components of their computer are dropped off when their computer is intaken by the Aide Station. When they pick up the machine, they sign another form acknowledging that repairs are completed and all components are being returned. These forms contain all of the relevant information from the ticket.

The Helpdesk needed a way to print these receipts. Since the software is a web application, one possibility was simply to spawn another browser window displaying a "printer friendly" version of a ticket (without a header, navigation bars, etc.). The Helpdesk operator would then print the ticket on the Aide Station's networked laser printer by using the browser's print function. Unfortunately, this would not have been adequate for several reasons: it takes time to open another browser window and print the document on a networked laser printer; the laser printer may be busy with other print jobs at the time, slowing the intake process; and we wanted to give the customer a receipt for the work performed.

3.3.4 Automating customer backups

Another function that was becoming unwieldy with the increased number of tickets at the Aide Station was the management of customer file backups. If reinstalls or hardware repairs were performed on computers, the Aide Station had traditionally backed up customer documents in limited quantities to the Aide Station's departmental space on one of our NetWare file servers. With the increase in tickets, this system was becoming a managerial nightmare. People would forget to delete backups, causing the volume to fill up. Occasionally, current customer backups would be mistakenly deleted. The solution was to have the Helpdesk software manage the creation and deletion of directories automatically.

The solution we chose was a combination of time-tested traditional technologies. Two 24-pin impact dot-matrix printers (one for the Aide Station, another for Computer Repair) were retrieved from storage and outfitted with carbonless forms. To interface these with the Helpdesk system, we used Lantronix Multiprotocol Print Servers. We acquired two Lantronix boxes and configured them to only support UNIX LPR services. When ATOPS open or close a ticket in Helpdesk, there is an option at the bottom of the form to select a printer for the receipt. The web server then invokes a Perl script which formats the data and passes it to the UNIX LPR system for printing.

Drew has been doing Linux/NetWare integration for years, so the solution came naturally. A special account was created in NDS that was allowed to login only from the Helpdesk server and given full file rights to the customer backup directory. We used freeware ncpfs utilities to mount the NetWare volume to the helpdesk server, and then scripts within Helpdesk automatically create customer backup directories by ticket number whenever a computer is intaken. When an Aide Station operator attempts to close a ticket, the software first checks for the presence of backup files. If backup files exist, the operator is required to click through a warning dialog indicating that the files will be expunged. A nightly chron job on the helpdesk server then expunges customer backup directories for tickets that have been closed for more than five days.

4. EXAMPLE OF COMPUTER TRACKING FLOW

4.1 Creating a New Ticket

When a ticket is first opened, the operator is asked to enter either the ID number, username, or extension of the customer reporting the problem. This generates a form that includes all the information on that user and any assets owned by them or by their department. A drop-down list allows the operator to choose an asset associated with the user. At this point, all components being intaken with the computer should be selected.

If the asset being intaken is not one of the assets listed in the drop-down menu, there is an option of "Other, Please Describe in Problem Area". If this option is selected, the operator must list the type of computer and the serial number first in the problem description window, and then proceed to list the reported problem.

The last step is for the ATOPS to determine what type of problem is being reported. If it is an obvious hardware problem it can be intaken as such, but if the cause of the malfunction is in doubt, it should be intaken as a software problem and have diagnostics run on the system

4.2 Editing Existing Tickets

After being opened, a ticket may be worked on by any number of operators or full-time staff. Each time the ticket is accessed, it automatically date stamps the "Steps Taken" and "Internal Comments" fields and adds the username of whomever is logged in to that session. These changes are not saved unless the ATOPS clicks the "Update this ticket" button.

There are seven status flags that can be associated with a ticket. They are:

- Open - Still in diagnostics, also used for known software problems.
- Hardware - It has been diagnosed with a hardware problem; the Hardware Support Specialist will further test the hardware before deciding it needs to go to the manufacturer for warranty repair.
- Sent Out - The machine has been sent back to the manufacturer for repair.
- Parts - Parts are on order for this machine.
- Pending - Waiting for contact from the customer. A comment should be added indicating what we need from the customer.
- Waiting - The computer is fixed and is waiting for the customer to pick it up.
- Closed - The machine has been picked up by the customer, who signed the receipt acknowledging that all components were returned.

In addition to basic training on how to use the new Helpdesk software, this represented the first time we offered customer access to our diagnostic and repair process. We had to ensure all comments in "steps taken" were of an appropriate nature. It had become common-place for ATOPS to vent frustration about customers/problems when we were using the internal DEC/NOTES system. The new system required a shift in behavior to a more professional support model. We realized that there are times when comments on a customer's disposition or tolerance level are appropriate and decided to make use of the "internal comments" section of each ticket for this purpose. The students appreciated the ability to post comments internally that wouldn't be available to customers.

6. CONCLUSION

5. IMPLEMENTATION ISSUES

5.1 Populating the Database

Once the system was tested and proven to be working according to the necessary specifications, we still needed to populate the database with the asset information for students, faculty, and staff. The incoming freshman data posed a problem as their machines would not be assigned to them until they were physically on campus. We were unable to access live data from our Administrative Computing department, which maintains a strict firewall and secure access to university data. Since our database structure does not match theirs anyway, they sent us raw data and we used Perl scripts to reformat the initial data feed that was supplied to us.

5.2 Interface Design and Testing

No matter how well-written a program is, it will not be used if the interface is not clean and easy to use. After loading the appropriate data into the system, there was an obvious need to clean and organize the user interface. We assigned one of our student managers to this task. She worked with the PHP and Perl scripts using her advanced skill in writing clean, readable HTML code and logical form design. The work she provided allowed the Helpdesk functions to be intuitive and easy to use. One of the more popular features in the Helpdesk interface is the ability to customize each user's browser colors, name, and signature line in the Phorum conferencing area.

5.3 Training

We have been using the Helpdesk software successfully for almost a year, intaking more than 800 computers, monitors, printers, network interface cards, and various other peripherals. We have identified specific problem trends with certain models, reduced the amount of misplaced equipment, and improved turnaround time. We plan to update the software over the summer of 2000 to include a loaner pool module where we can assign loaner computers to customers whose computers are at the manufacturer for an abnormally long time. We also plan to create printing and scanning services tickets to track other services performed by the Aide Station and more closely track individual assets and inventory as part of a package.

The new version will also phase out our reliance on Administrative Computing for data by moving the asset information into our own inventory control database, ensuring up-to-date and accurate data .

7. REFERENCES

[1] Yarger, Randy Jay, Reese, George, and King, Tim, *MySQL & mSQL*, O'Reilly & Associates, Sebastopol, CA (July 1999).

[2] Atkinson, Leon, *Core PHP Programming*, Prentice Hall. (May 1999).

[3] http://www.phorum.org

[4] http://www.php.net

[5] *MySQL Reference Manual for version 3.23.21-beta*, (1999) http://www.mysql.org/Manual/manual_toc.html