Netlabyrinth: an Intranet Application for Tracking Support and Network Resources

By

Thal La Charity

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfilment of the Requirements for
the Degree of Bachelor of Science
in Information Engineering Technology

University of Cincinnati
College of Applied Science

June 2004

Netlabyrinth: an Intranet Application for Tracking Support and Network Resources

By

Thal La Charity

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfilment of the Requirements
for
the Degree of Bachelor of Science
in Information Engineering Technology

_____	_____
Thal La Charity	Date
_____	_____
Prof. Russell McMahon, Faculty Advisor	Date
_____	_____
James F. Sullivan, Department Head	Date

Acknowledgements

I would like to give special thanks to Tom Wulf, former Network Administrator at the University College Computer Network and current professor at the College of Applied Science, for finding a real-world need and allowing me the opportunity to bring a solution to fruition. I would like to give special thanks to Eric Fulkert and Bill Houk, current owners and proprietors of Prospera Solutions Group, for supporting me through the final development stages, providing a constructive work environment, and addressing my hosting and testing needs. I would like also to give special thanks to Jerry Felix and the staff at *ec link* for providing an excellent learning environment to hone my ASP coding skills. Thanks are also due to my mother Linda and father Ralph for, among other things, their continued support of my educational endeavors. Finally, I give thanks to my wife, Yvonne La Charity and my children, Devin and Sasha, for their love and patience.

Table of Contents

List of Figures

Abstract

Netlabyrinth is a web-based intranet solution designed as a means of keeping track of computer assets and to organize efforts in maintenance on a network. Users include network administrators and computer technicians who need accurate network asset data. Users have the ability to receive and report on network support issues while on the move across a network. *Netlabyrinth* is a continually evolving technical tool created using tried and true development paradigms: Hypertext Markup Language (HTML) enhanced via Active Server Pages (ASP) with a SQL Server back end. *Netlabyrinth* brings up to the minute SQL Server-driven network information to your fingertips through an intuitive web interface.

Netlabyrinth: an Intranet Application for Tracking Support and Network Resources

1. Statement of the Problem

This project was first conceptualized after speaking with network administrators and network technicians at the University College Computer Network or UCCN and other administrative and technical figures of small businesses(1).

UCCN is a computer network of over five hundred workstations within University College at The University of Cincinnati. The college has equally complex and extensive computer labs and faculty office networking.

UCCN staff believed that they would likely benefit from a system that would allow for the tracking and reporting of support issues and resolutions. One plan that was in the works was to add a new internal application for the tracking of network hardware resources. This plan was preceded by a legacy Cold Fusion application called Tech Tracker that had long since grown slow in comparison to competing technologies. Tech Tracker was no longer being used and was effectively defunct.

2. Description of the Solution

After the abandonment of Tech Tracker, UCCN staff were essentially reduced to using pen and paper or email for the logging of Support issues. It was decided that a new, more robust replacement for Tech Tracker was needed to make resource and support tracking more efficient. The new application to be developed would emerge as *Netlabyrinth*. *Netlabyrinth* would be the faster, more intuitive cousin to Tech Tracker.

Advances in database technology have provided an excellent, high-performance data storage solution in Microsoft SQL Server 2000 combined with the solid web technologies of Microsoft's IIS family of web server software(2). I used

my knowledge of web design, the SQL query language, and my knowledge of Active Server Pages to create *Netlabyrinth*, a system that would accomplish the following:

- Allow entry of Support issues as they happen, from any network-connected workstation.
- Allow administrative and technical staff to view new support notices as they are inserted into the SQL Server database.
- Facilitate the ability of technical staff to accept support jobs as they are made known.
- Enable technicians to report on the status of their pending support jobs from start to finish.
- Enable network administrators to view jobs at any stage of their life cycle.
- Keep track of workstation statistics and hardware resources across the network.

Netlabyrinth is intuitive for IT professionals to use and has navigational buttons that describe the usage capabilities clearly and depress when selected. The button effects were achieved with the use of Adobe Photoshop and Dynamic HTML(3). The primary usability goal was to result in an Intranet site with basic authentication capabilities and simple to follow application flow.

Netlabyrinth is the convergence of two application designs. One of those designs is a second look at a preexisting hardware resource tracking application with modifications to make the new incarnation function more efficiently. The other design is a fully customized interface that allows for the tracking of Support issues. The designs mesh together to form the new product that is *Netlabyrinth*.

2.1 User Profile

Netlabyrinth is designed for two groups of users with differing levels of oversight on a given network: network administrators who oversee Support resolution and network technicians who embark upon the completion of Support issues. Both

groups should have requisite knowledge of web browsers and web browsing. The overall design is covered in the next section.

The product is primarily targeted as a productivity tool for network technicians and as such, technicians are the group that will gain the most benefit from *Netlabyrinth*. Technicians can easily view their assigned jobs and perform written updates as the status of their jobs. Technicians can also view and update the hardware specifications of any given workstation on the network.

Netlabyrinth also targets network administrative staff and provides advanced network overview abilities. Administrators are granted the ability to view macro-level reports provided by links to a Crystal Reports Report Application Server, connected to the SQL data store (4). Administrators are also able to see completed Support jobs for better management of their IT resources.

2.2 Design Protocols

The main window of the application is generated using HTML and ASP as designed from scratch using a combination of a simple text editor, Notepad, and Microsoft's web development platform Visual Interdev. The common navigation at the top of each page is contained within an include file that is coded in ASP to generate a header to each page each time it is invoked. The main window is also 800X600 pixels in size, the most likely resolution a user's workstation will be set for (5).

Each section of the *Netlabyrinth* application is customized to the function that a given page serves. Each page utilizes the include file mentioned above, but the remainder of the page is coded in ASP to serve specific uses such as displaying active Support jobs or a listing of the network's connected workstations. Most of the ASP development took place within Visual Interdev because of the comprehensive and

structured way it displays ASP code contrasted with the encapsulating HTML code (6).

For the login screen, an HTML page passes login information to an intermediate ASP page. The intermediary active server page processes the login data via ADODB connections and record sets that interact with data in the SQL Server back end (7). The level of security provided by this method of authentication is adequate for the purposes of an internal Intranet application that has no outside Internet access configured.

Microsoft SQL Server 2000 provides excellent performance and superior reliability compared to smaller, lighter databases such as Microsoft Access (8). By default SQL Server is more secure and has by far more security features than Microsoft Access. Automated backups and event logging are two features of SQL Server that alone make it the de facto database choice among Microsoft offerings.

The starting page of the application is the *Netlabyrinth* login screen which redirects and logs unsuccessful or incorrect logins. When a correct set of authenticating criteria are entered, the main window for the application appears and the login screen then becomes a logout screen in the background. The main window immediately presents the user with a screen that awaits entry of new Support issues. The user can also proceed to four different sections of the application via the common navigation bar: New Issue entry, Current job listing, Categorical job search, and Workstation listing. An overview of the application to this point can be seen below (See Figure 1.).

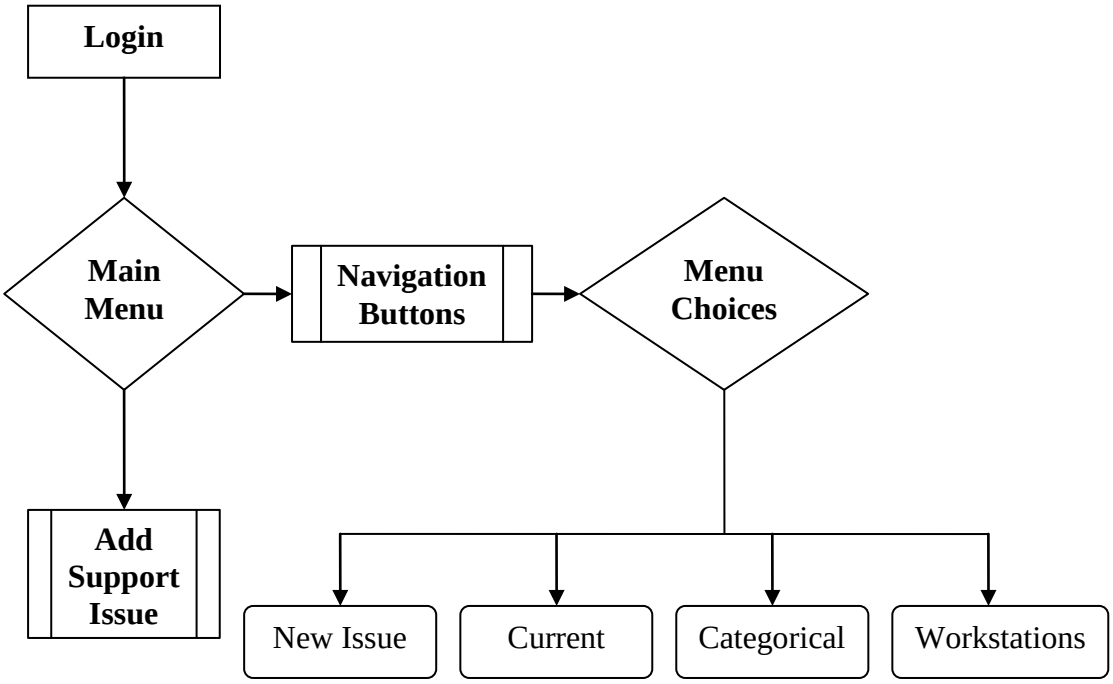

Figure 1. ***Netlabyrinth*** **Overview**

From the initial main window location, one is able to select options from the top menu bar (See Figure 2.), the first of which is the New Issue button. On the initial screen however, the user is already immediately placed in the interface for entering new Support records. The fully expanded New Issue interface can be seen below (See Figure 3.).

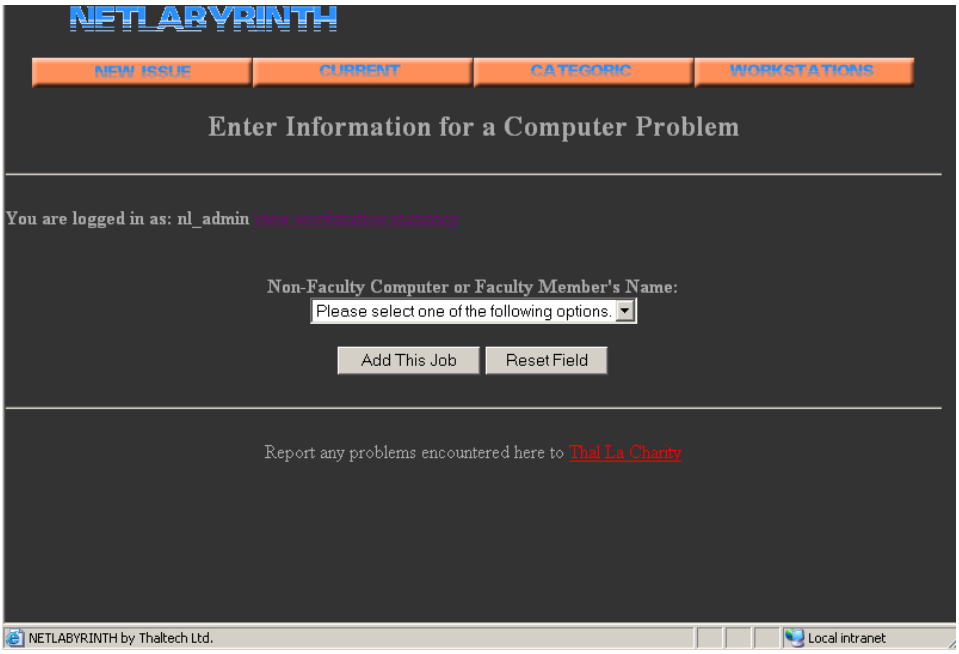

Figure 2. *Netlabyrinth* Initial Main Window

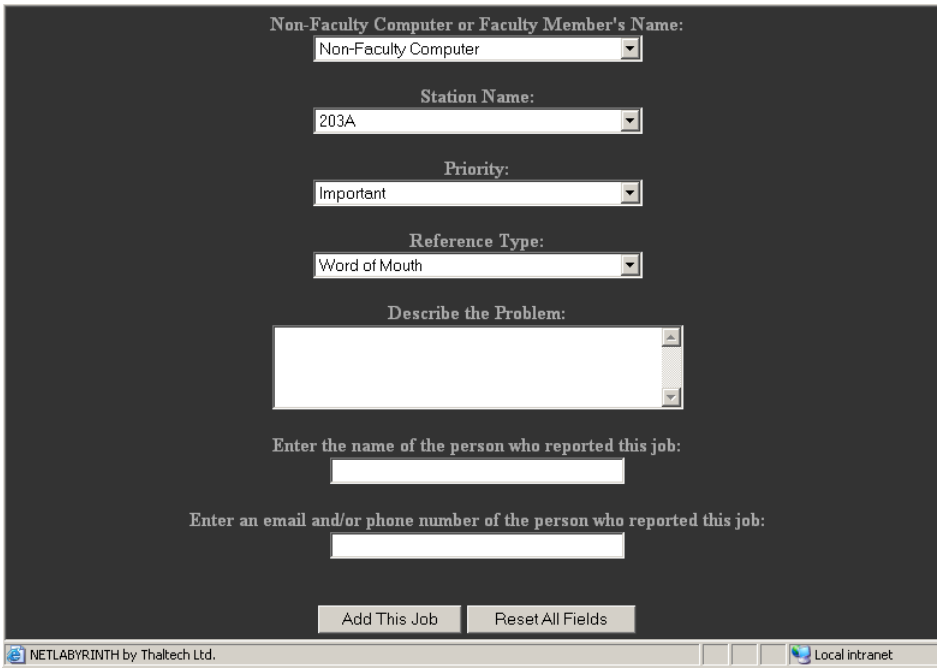

Figure 3. *Netlabyrinth* Support Entry

The next selection on the main menu is the Current button. The Current button takes the user to a page that lists all of the currently available and unassigned Support issues. In the Current window, a technician can see queued jobs and select from a list box to accept and begin to confront Support issues. See the Current window below (See Figure 4.).

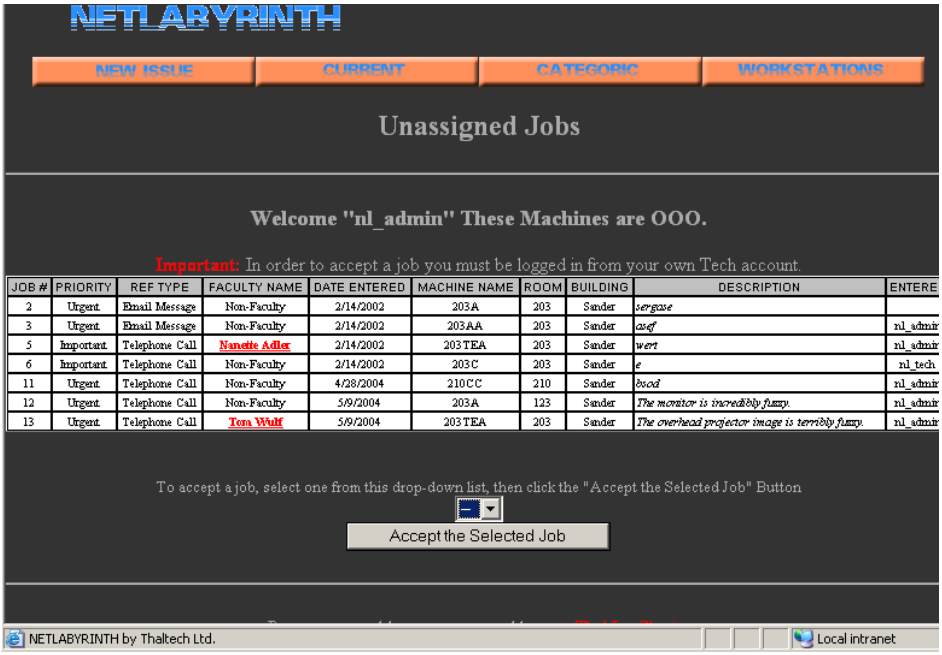

Figure 4. *Netlabyrinth* Current Support Issues

The next main menu selection is the Categoric search button. The Categoric search button takes the user to a page that facilitates search capabilities by faculty member or by workstation ID. If a job is open then it can be sought from one of the list boxes on the Categoric section. The Categoric search page can be seen below (See Figure 5.).

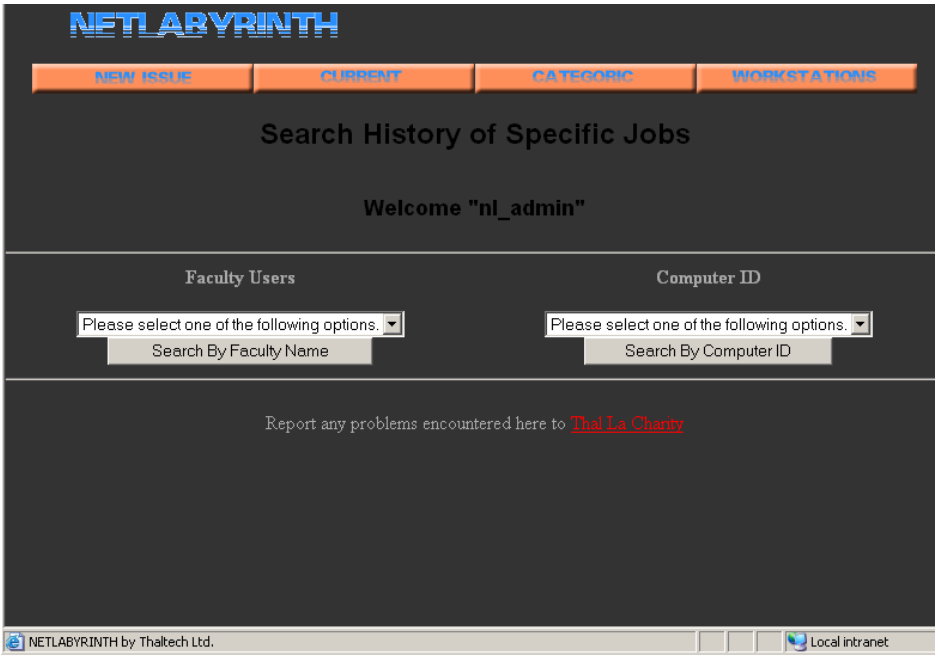

Figure 5. *Netlabyrinth* Categoric Search of All Support Issues

The fourth and final selection that can be made from the main menu is the Workstation button. The Workstation button takes the user to an updated version of the legacy Tech Tracker application. In the Workstation window, all workstations on the network are listed. Each workstation record is editable and a staff member is able to add new workstations as they are implemented on the network. The Workstation page is detailed below (See Figure 6.). The interface that comes up when a workstation is selected can also be seen (See Figure 7.).

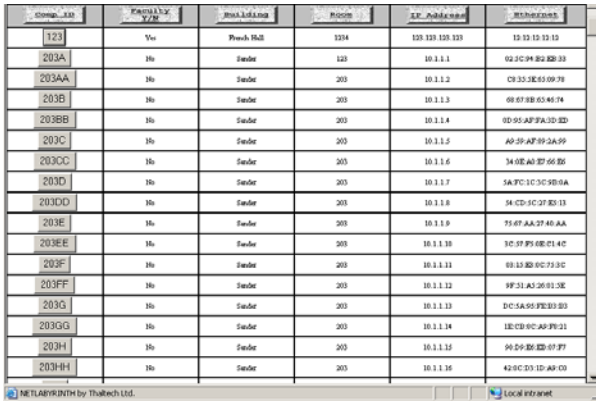

Figure 6. ***Netlabyrinth*** **Workstation Listing**

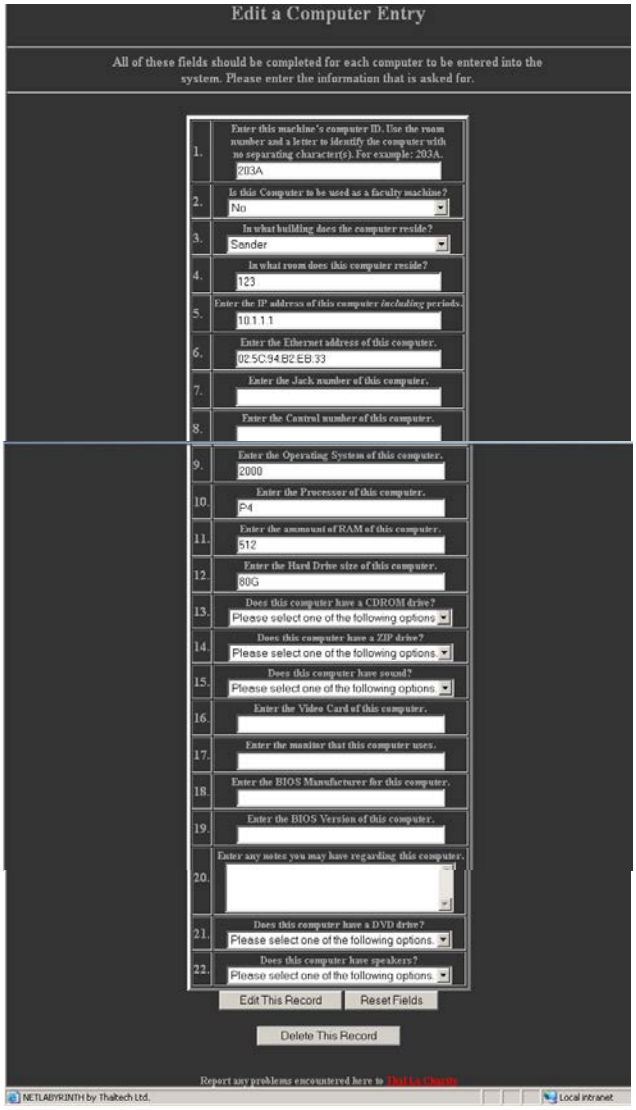

Figure 7. ***Netlabyrinth*** **Workstation Edit & Delete**

3. Deliverables

To provide a more intuitive and logically flowing application, various modifications were deemed beneficial. During the design phase of *Netlabyrinth* the following deliverables were defined:

- Development of intuitive navigation
- Development of authentication system
- Further development of the technical support area
- Further development of the workstation tracking area

4. Design and Development

What follows includes the project's timeline, total budget including hardware, and written reference cost.

4.1 Timeline

The development of *Netlabyrinth* came with a gamut of bug fixes, design considerations and reconsiderations, and the realization of met objectives.

4.1.1 Senior Design I Accomplishments

During Senior Design I, the following items were accomplished:

- Investigated legacy application for inclusion of functionality
- Researched the IT tracking needs of a sizeable network
- Learned the syntax and programming conventions of ASP
- Assembled mock pages for each of the application areas in HTML
- Developed my proposal and oral presentation

During the research and discovery phase of *Netlabyrinth* I studied and dissected the legacy application Tech Tracker for a development foundation. As a former UCCN technician myself, I was able to piece together basic requirements for a system like *Netlabyrinth*. I also spent time speaking to network administrators and technicians about how the new system could best benefit them. Prior knowledge of networking, technical support, and gathered feedback was instrumental in the design of *Netlabyrinth*.

4.1.2 Senior Design II Accomplishments

During Senior Design II, the following items were accomplished:

- Designed an Access database with web connectivity
- Developed functional ASP code to drive the existing interface
- Started developing of an upsized SQL Server database back end

- Prepared Design Freeze documentation and oral presentation

Implementing a fully functional web application took less time than expected as much of the interface was already in place, only requiring the addition of ASP enhancements. Now that I had a functioning application, it could be tested. Several technicians participated in the alpha testing and bug reporting to get *Netlabyrinth* to a demonstrable status. Aesthetic and authentication concerns had not been sufficiently addressed by this point.

4.1.3 Senior Design III Accomplishments

During Senior Design III, the following items were accomplished:

- Completed SQL Server database back end
- Performed testing of the revamped design layout
- Performed intensive bug fixing
- Tuned the application's flow
- Presented the final project

During testing of *Netlabyrinth*, it was found that the application ran very efficiently and with better than expected performance on a variety of platforms. The fact that all database processing can be carried out on a dedicated SQL Server implementation reveals that the processing load on the client is minimal (9).

4.2 Budget

The budget for *Netlabyrinth* can be seen below (10).

Hardware	
Item	**Cost**
Machine 1 Estimated Value	$1000.00
Machine 2 Estimated Value	1200.00
Total Hardware:	$2200.00
Software	
Item	**Cost**
MS Windows 2000 Professional	$499.99
MS Office 2000 Professional	210.00
MS Visual Studio 6.0 Professional	665.00
MS SQL Server 2000 Standard 1 Proc	1150.00
Adobe Photoshop	600.00
Total Software:	$3124.00
Grand Total:	**$5324.00**

Figure 8. ***Netlabyrinth*** **Budget**

5. Proof of Design

This section describes the process of deliverable implementation and some challenges that were encountered along the way.

5.1 Navigation

Navigation functionality can often make or break an application. It was important to have uniform navigation across the entire application so coming up with a way to keep the navigation code in one place was essential. Deciding on the use of an include file that is used throughout the application proved a successful means of confronting this issue (11).

5.2 Authentication

Authentication, which had until now been nonexistent, was developed and added to the start of the application. The user's name and password are now authenticated against user data in the SQL Server database. Successful and unsuccessful login attempts are logged to a table in the database.

5.3 Technical Support Reporting

Entry of technical support jobs is now a smooth flowing process compared to earlier versions of *Netlabyrinth*. The data entry has concisely defined fields present on the data entry screen. Updates in status are accomplished via memo fields for each step in the job completion process and follow up or conclusion reporting is required before the completion of any Support job.

5.4 Workstation Tracking

The workstation tracking component of *Netlabyrinth* provides access to all data regarding networked workstations. Each column of data has a button that allows that column to be sorted in ascending or descending order. Each row, representing an individual workstation, includes a button that takes the user to an editing screen for

the computer on which the user may update or delete existing workstations. New computers can be added from the main listing page.

5.5 Crystal Reports Statistics

Through the power of Crystal Reports Report Application Server, administrative users are able to see various macro-level charts and reports. Crystal Reports meshes well with data stored in SQL Server and provides an invaluable tool to higher level network staff that needs to see regularly updated network statistics.

6. Testing Procedures

Netlabyrinth version testing was a necessary part of the development process. All code was tested upon completion of large segments of ASP programming. Also, usage testing was performed by technicians after my initial testing was complete. Code fixes were relatively easy to track down and eliminate especially through external usage testing.

There is no more effective means of testing than to place the application in a real world environment and have the target users attempt to apply *Netlabyrinth* as it is meant to be used. Often a developer will do good initial testing, but when it comes to how people will actually use the application, there is no substitute for real world testing.

Netlabyrinth was tested with Windows 2000 Professional and Windows XP Professional operating systems, using Internet Explorer 5.x and better and Mozilla 1.6 web browsers.

7. Conclusions and Recommendations

7.1 Conclusions

Netlabyrinth was developed in response to UCCN's need for an intranet based application that handles network reporting and better support accountability. As a network reporting tool, *Netlabyrinth* meets the need by allowing for the entry of detailed workstation information. As a technical support tool *Netlabyrinth* is effective at using multi-stage reporting. All tools that were used in the development of *Netlabyrinth* were appropriate and met or exceeded all expectations.

7.2 Recommendations

When developing web applications of all kinds, it is important to remember that good performance is essential. One stumbling block to early *Netlabyrinth*

development was being initially limited to the use of Microsoft Access as a database. While Microsoft Access is an excellent learning tool, sufficient for small, non-critical applications, it has limitations including the fact that it runs slower than SQL Server and lacks the scalability that SQL Server provides. This was remedied once the Access database was upsized to SQL Server 2000. While Access is recommended as an introductory database learning tool, SQL Server is the far superior choice when implementing database-driven applications in a real world environment (12).

Another limiting factor encountered during *Netlabyrinth* development was that I had to use legacy desktop computers as web and database servers. Much like the Access example above, these lower end computers were good for learning and development purposes, but when it came to actual implementation, performance suffered greatly during real-world use. Dedicated, server-class computers proved to be far more appropriate. Deployed applications benefit greatly from the combination of higher end database engines and higher end hardware.

Equally important as the database and hardware, is the choice of development tools used in the production of custom applications. ASP, hosted from a dedicated server, was an excellent choice during the majority of the development process. However, over time it has become apparent that ASP.NET is the wave of the future and eventually *Netlabyrinth* will require being ported to the newer technology (13). It is recommended by the author that future development be performed to that end.

Appendix A.

Research Information

Much or my research was conducted via the creation of ASPs for co-op employers. I have found that the best sort of research is only accomplished with real-world requirements driving them. For example, my first successful ASP was a faculty and staff index/listing created for UCCN. It drew information from an Access database and displayed contact data for faculty and staff members. This, coupled with various programming reference materials, was instrumental to my introduction to the world of data-driven web programming.

Other extremely useful research has always been drawn from the knowledge of my programming peers. Working in a team environment provides a knowledge base that cannot be recreated in any reference book or classroom. Being able to ask questions of and answer questions for peers is an excellent means of gathering experience.

Appendix B.

Project Timeline

Task	Start	End
Senior Design I	*01.03.01*	*03.17.01*
Feasibility	01.10.01	01.16.01
Report	02.11.01	03.05.01
Presentation	03.12.01	03.16.01
Senior Design II	*06.17.01*	*08.29.01*
Prototype	06.18.01	08.25.01
Access Database Dev.	06.18.01	06.28.01
Tech Support ASP Frame.	06.29.01	07.15.01
Workstation ASP Frame.	07.16.01	08.01.01
ADO Connectivity	08.01.01	08.07.01
QA/Flow Testing/Tweaking	08.08.01	08.25.01
Report	07.05.01	08.08.01
Revisions	08.23.01	08.25.01
Presentation & Preparation	08.15.01	08.29.01
Senior Design III	*04.01.04*	*06.03.04*
QA Testing & Enhancing	04.06.04	05.26.04
Upsize to SQL Server database	04.02.04	04.03.04
Enhanced Navigation	04.12.04	04.28.04
User Authentication	05.01.04	05.07.04
Report	05.08.04	05.17.04
Presentation	06.03.04	06.03.04

Appendix C.

Code Snippets

C 1. Call to Navigation Include File

Establishing a means of maintaining a common menu system throughout the development of *Netlabyrinth* was vital to effectively creating a useful user interface. This was accomplished, as stated previously, via the use of an include file that contained the header portion of each page within the application. The code that invokes the include file and is found at or near the head of every ASP is as follows:

```
<!-- #include file="./myincs/nav.inc" -->
```

C 2. Contents of nav.inc

Within the include file, nav.inc, is code used for preloading images. The code allows *Netlabyrinth* to load much faster than it otherwise would.

```
<script language="JavaScript1.2">
        <!--
        javascript:window.history.forward(1);
        var preload='false';
        var pic = new Array();
        for(i=1;i<=17;i++) pic[i] = new Image();
        pic[1].src = '../ttnew/images/netlab.png';
        pic[2].src = '../ttnew/images/wsu.png';
        pic[3].src = '../ttnew/images/wsd.png';
        pic[4].src = '../ttnew/images/niu.png';
        pic[5].src = '../ttnew/images/nid.png';
        pic[6].src = '../ttnew/images/cuu.png';
        pic[7].src = '../ttnew/images/cud.png';
        pic[8].src = '../ttnew/images/cau.png';
        pic[9].src = '../ttnew/images/cad.png';
        pic[10].src = '../ttnew/images/xtu.png';
        pic[11].src = '../ttnew/images/xtd.png';
        pic[12].src = '../stationtracker2/images/compid.jpg';
        pic[13].src = '../stationtracker2/images/facyn.jpg';
        pic[14].src = '../stationtracker2/images/building.jpg';
        pic[15].src = '../stationtracker2/images/room.jpg';
        pic[16].src = '../stationtracker2/images/ip.jpg';
        pic[17].src = '../stationtracker2/images/enet.jpg';
        preload='true';
        function chgImg(daImage,id) {
```

```
                id='pic[' + id + ']';
                if (preload) {
                        newImg = eval(id + '.src');
                        daImage.src = newImg;
                }
        }
        // -->
</script>
```

References

1. Wulf, Tom. Personal interview, Lab Director. January 4,2001.

2. Trent, Rod. *Windows 2000 IIS 5.0: A Beginner's Guide*. California: McGraw-Hill, 2001.

3. DynamicDrive.com. Http://www.dynamicdrive.com. 2001.

4. Peck, George. *Crystal Reports 9: The Complete Reference*. California: McGraw-Hill, 2003.

5. Wpdfd.com. Http://www.wpdfd.com/wpdtame.htm. 2004.

6. Amundsen, Michael. *Using Visual Interdev 6*. USA: Que. 1998.

7. Asp101.com. Http://www.asp101.com. 1999.

8. Mssqlcity.com. http://www.mssqlcity.com/Articles/Compare/SQLvsAccess.htm. 2004.

9. Wynkoop, Stephen. *Using SQL Server 7.0*. USA: Que. 1999.

10. Google.com. "Smart Shopping throught Froogle". Http://www.google.com/froogle?hl=en&tab=wf&q=. 2004.

11. W3schools.com. http://www.w3schools.com/asp/asp_incfiles.asp. 2004.

12. Microsoft.com. http://www.microsoft.com/resources/documentation/sql/2000/all/reskit/en-us/part2/c0561.mspx. 2004.

13. Payne, Chris. Sams Teach Yourself ASP.NET in 21 Days. Indianapolis: Sams Publishing, 2001.