# Automated Request and Document Management Systems

By

Stephanie Pero

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

University of Cincinnati
College of Applied Science

December 2005

# Automated Request and Document Management Systems

by

Stephanie Pero

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



| | |
|---|---|
| Stephanie Pero | Date |
| Hazem Said, Faculty Advisor | Date |
| Patrick C. Kumpf, Ed.D. Interim Department Head | Date |

# Acknowledgements

I would like to give special thanks to Belcan Corporation and specifically to my direct manager Rick Perazzo for helping me through all the corporate “red tape” and working with me to finish my project ahead of schedule. I would also like to thank Roger Bowman for being my technical support throughout this project. If I couldn’t figure something out he was always there to help point me in the right direction. I would also like to thank Dr. Said for standing behind me 100%. He made this a challenging but significant quarter. Without Dr. Said’s guidance I would not have had the opportunity to graduate a quarter early. Finally, overall thanks and appreciation to everyone who was involved with this project, you are the ones that made it a reality.

# Table of Contents

# List of Figures

**Figure Number** **Page**

# Abstract

Belcan Corporation has used paper forms to handle employee's request to the Computer Services department since the creation of the company. This manual system has many flaws, of which is the loss of the paper trail especially with the form level of approval required for every request. This Web application was built using java technologies and designed for easy use by all users. With a user friendly interface anyone with the most basic knowledge of computers will be able to use this system. This also provides a fast solution to what used to take weeks to accomplish. The system makes use of four forms, but has the room and technology to expand. This system will allow for all users to easily access and submit a request to the Computer Services team, and to track the status of their requests. Along with the forms, the system offers a service that allows for users of all types to view all requests that they have submitted. In addition, the application enables the technicians to address a request. Overall, the Automated Request and Document Management Systems will allow the IT staff at Belcan to better support all their users.

**Automated Request**
**And**
**Document Management Systems**

## 1. Statement of the Problem

Belcan Corporation was founded in 1958 by Ralph G. Anderson, who is still heading the operation today. Belcan offers Full-Service Engineering, Design and Build, Application Technology, Procurement, Information Technology, Technical and Temporary Staffing, and Multimedia Services worldwide. Belcan Corporation is headquartered in Cincinnati, Ohio and provides services in several state and many foreign countries through a network of locations including worldwide alliances. Belcan has annual sales of $300 to $400 million and employs roughly 4000 employees across the globe. (9) With this background information, the considerable size of this company is apparent, and with a company of this stature, one would expect that the way of business is conducted in such a way that is cutting edge. This is far from the truth.

"Most IT organizations still install and maintain computers the same way the automotive industry built cars in the early 1900's: An individual craftsman manually manipulates a machine into being, and manually maintains it afterward. This is expensive. The automotive industry discovered first mass production, then mass customization using standard tooling." (10) So why doesn't the IT business do the same? Well many businesses have implemented the use of fully automated systems and have standardized the practices and tooling that is needed to do the same within IT. This type of standardization allows for all paper trails to be electronic rather than actual hard copies of documents, along with many other uses. This is where this project came into fruition.

Belcan deals with all of their computer related requests with paper copies of forms that are filled out by a user then sent off to the appropriate people to get signed and approved. This process takes weeks and is inefficient. There are numerous versions of forms that are being used by the users and there is hardly any documentation for reoccurring problems. This leads to confusion, incorrect requests, and ultimately a very unhappy customer.

With Belcan's continuous growth around the world it would be so much easier if an automated system was implemented to allow for Belcan's Infrastructure Team (Computer Services) to process these requests. This would allow for faster turnovers for all requests made and it would happen with little inefficiency. In addition, it would create a better working environment for all of Belcan's employees, which would boost the morale of Belcan's employees.

## 2. Description of the Solution

### 2.1 Product Description

The Automated Request and Document Management System is designed to replace the paper solution. The application is a three layer process that starts with the employees filling out a request form via the intranet and submitting this form online. The information is then passed onto the second layer which is the approval layer in which the approval is obtained from the different management levels. If the request is approved it is then sent off to the third and final layer, the technician. At this point in the process the request must be implemented and closed out. Once this happens the user is then notified that the request has been completed. The form is transferred from one level to the next

through an email notification system and automated application that updates itself according to the action taken. Figure 1 is a use case diagram that shows the different components of the system.

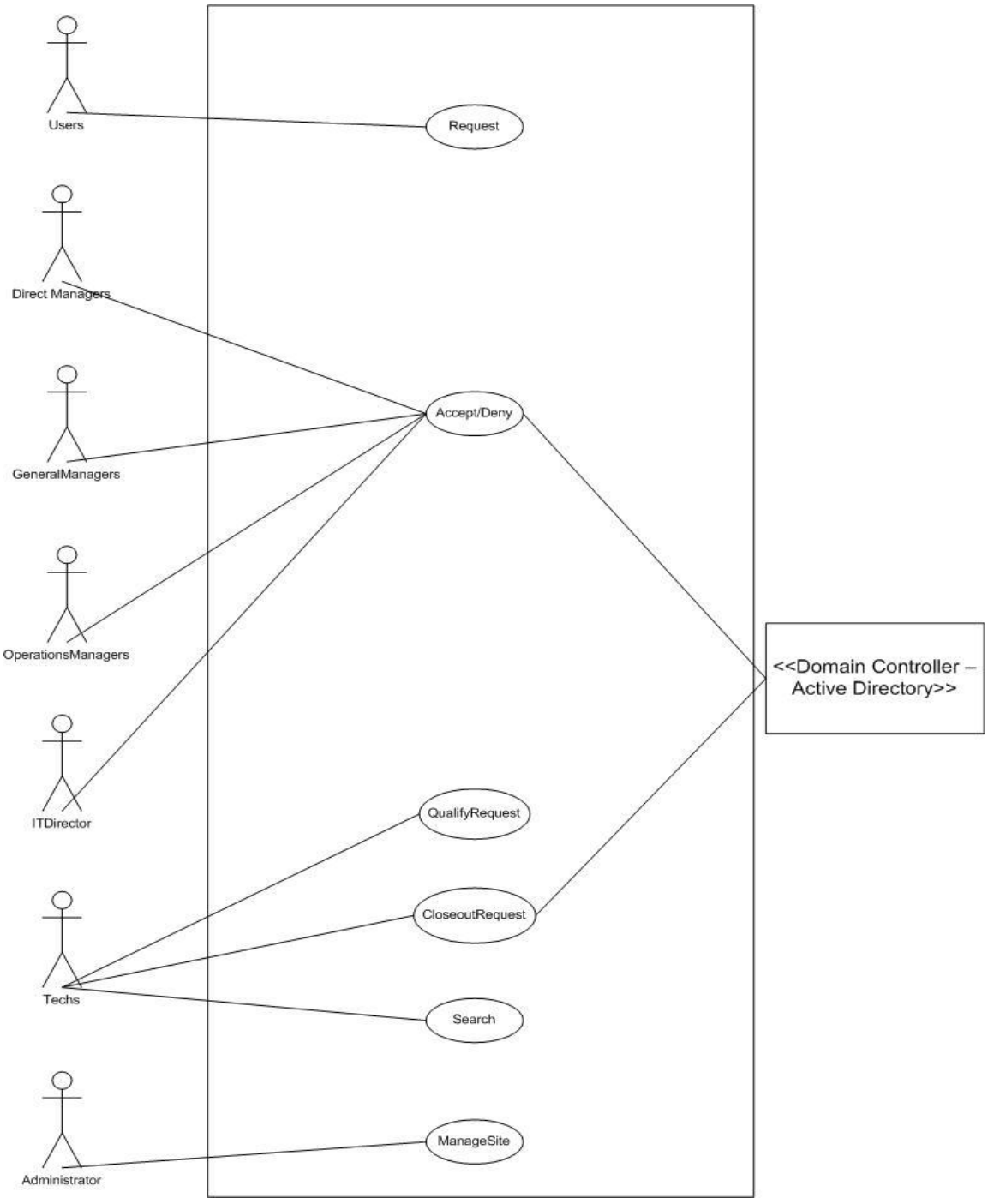


**Figure 1: Use Case diagram for The Automated Request and Document Management System**

The system uses a three layer architecture with a user-interface layer, a business layer and a data layer as shown in Figure 2.

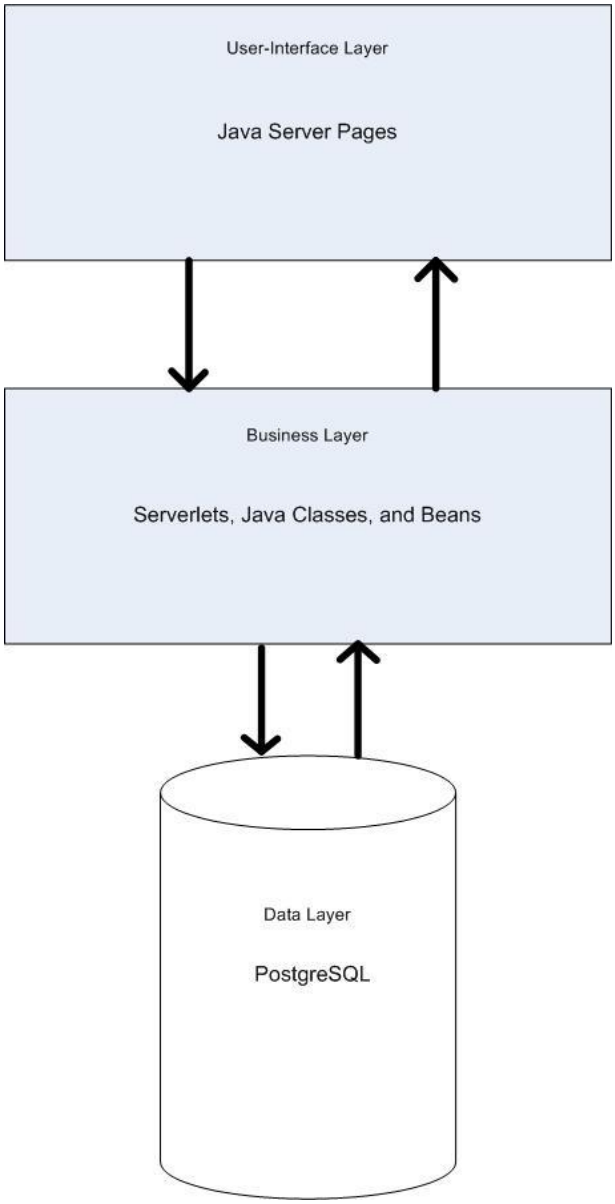


**Figure 2: Solution Architecture**

PostgreSQL database management system is used as the data repository while java server pages technology is used for the user-interface components. The business layer components are made up of Java Classes, Servlets, and Java Bean Technologies. The entire application is then hosted by an Apache Tomcat server.

### 2.2 Intended Use

The Automated Request and Document Management Systems are intended for Belcan employees. Belcan's Computer Services group offers many services via their intranet, but requesting items such as software, peripheral hardware, a new computer, or even a service is still all done with paper forms.

The new Automated Request and Document Management systems allow for employees to get onto the intranet, access and submit a form and dynamically send it off to their direct manager and on up the hierarchy of the company for approval which includes the General Manager of the division, the Operations Manager of the office, and the IT Director. Once this has gone to the top or to the IT Director, a technician is then sent the information so that they can implement the request and closeout it out.

During this process of approvals the employee is given access to check on the status of any request, which is dynamically changed. Allowing for Belcan employees to access such valuable information and giving them such a quick response, will result in a better working attitude and a more confident employee.

Each of the different processes is fully described in Use Case Scenario's. This enables for a quick understanding of what the system does, how it does the certain tasks, and who is able to do these tasks. It also offers a high-level understanding of what the

application is being built for. For an in-depth view of these scenario's please reference appendix A.

### 2.3 User Profile

There are technically seven different types of users. Each plays an important role in how this system functions. The seven include employees, direct managers, general managers, operations managers, IT director, technicians, and finally the administrators.

Belcan employees are mostly made of engineers. If something is not working or someone needs an item to enable them to work, Computer Services needs a solution that will get them back on track as fast as possible. If Belcan's employees can't work or access something then Belcan does not make any money. That just shows the importance of getting all of the employees up and running as soon as possible.

Each of the categories of managers (direct managers, general managers, operations managers, and IT director) plays an approval/denial role. This basically means that when a request is submitted these people interact with an interface that allows them to either accept the request or deny the request. Once the request has gone through each of these individuals another type of user is notified and the manager's role is finished.

The next type of user that is involved in the request process is the technician. The technicians are responsible for implementing the requests and finally closing them out. This group of people includes all of computer services and for each type of request that would be submitted there is a technician present that would take care of it.

The administrators of this system are fully responsible for the intranet and all applications that are created in house and are served through the intranet. The administrators are the final piece of this system, and the final type of user. Each of these users is advanced to expert computer users and experienced programmers.

### 2.4 Design Protocols

#### 2.4.1 Color Scheme & Outline

Belcan's intranet follows a strict color scheme which is made up of blue, black, gray, and some orange. In the main header, which is found on all of the pages in the site, Belcan's web development team has made good use of a menu bar and an image. The simplicity of this creates a professional look along with an easy to navigate site. The drop down menu's that are located within the navigation in the main header are all orange which make for a clean accent to the dark blue's and blacks. The site has a style sheet that is linked to it so that all of the links and headers are all the same on each of the pages. This ensures that the entire site follows the same outline and color scheme.

In the sectional header that is located just below the main header I have followed the color scheme and main presentation outline of the site by allowing access to the automated forms through a drop down menu. This not only follows the entire look and feel of the site but it makes the site easier to navigate through and allows for quick access to the forms. On the employee's home page of the application I made sure to follow all of the prior design outlines of the rest of the site. I did this to ensure that this section of the site fit in with everything else. An example of the color scheme can be viewed in Figure 3.

### 2.5 Users Interface

#### 2.5.1 Belcan Employee's Interface

The employee interface was created using Java Server Pages, Java Classes, and Java Beans. This is a web application, in order for the employees to access the interface they will need to log onto the correct section of the site. Once the employee has reached the correct section the employee will then have the ability to navigate through all of the previously submitted forms and check on the status or of the available forms via a dropdown menu. Figure 3 is an example of what the employee's interface looks like.

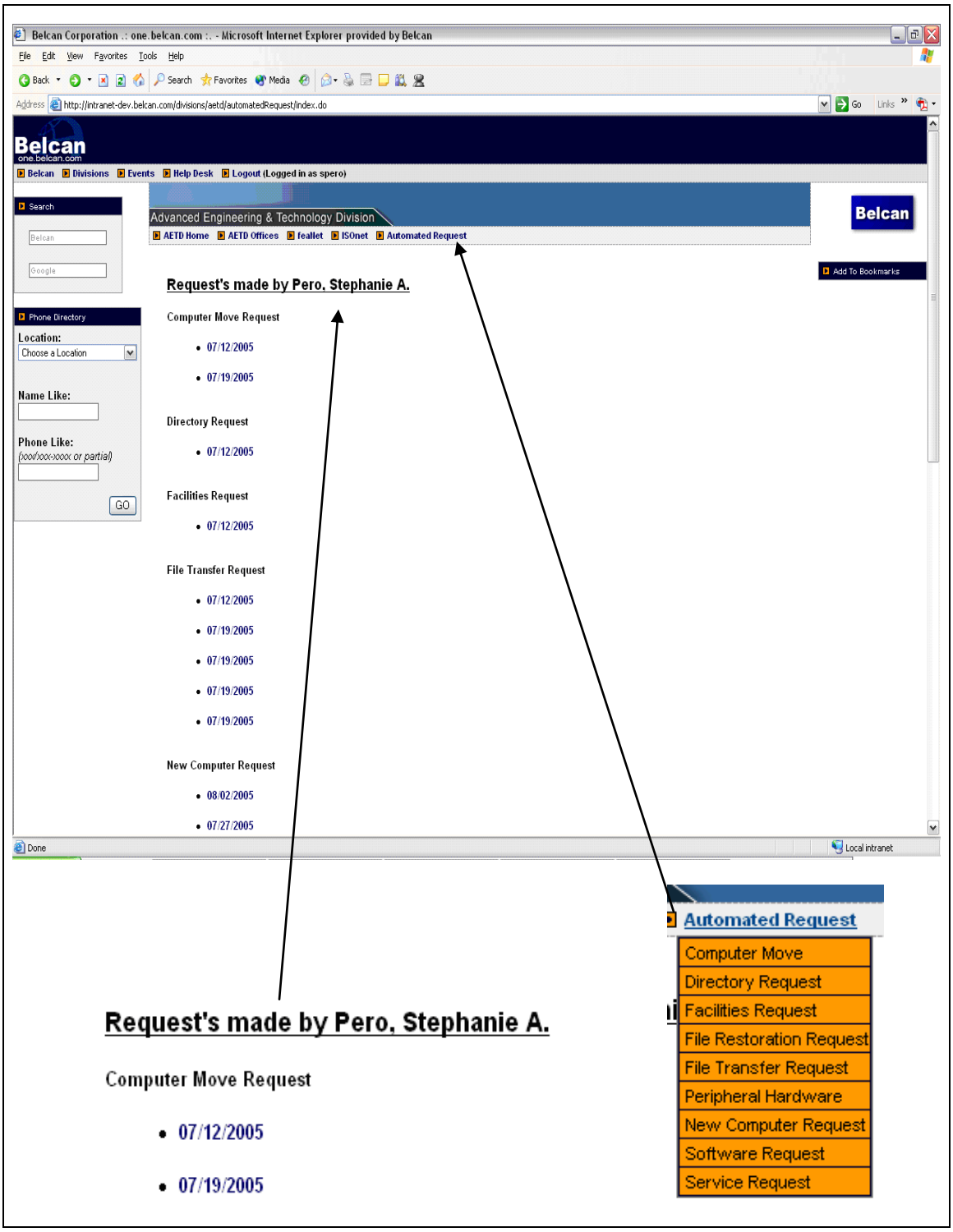


**Figure 3: Belcan Employee's Interface**

### 2.5.2 Managers Interface

Each set of managers (direct manager, general manager, operations manager, and IT director) will have the ability to view what requests are waiting for approval. This allows for fast and easy access to all request that have been submitted from their employees. The logged in managers access the requests through the links that are located on their home page of the application. Each of the links is displayed in sequential order of arrival by the date of the submitted request. Also, for quicker and easier identification each of the requests is sectioned by type of request. This allows for the manager to quickly retrieve a needed request for approval. Figure 4 represents what a manager would see upon logging into their home page of the application.

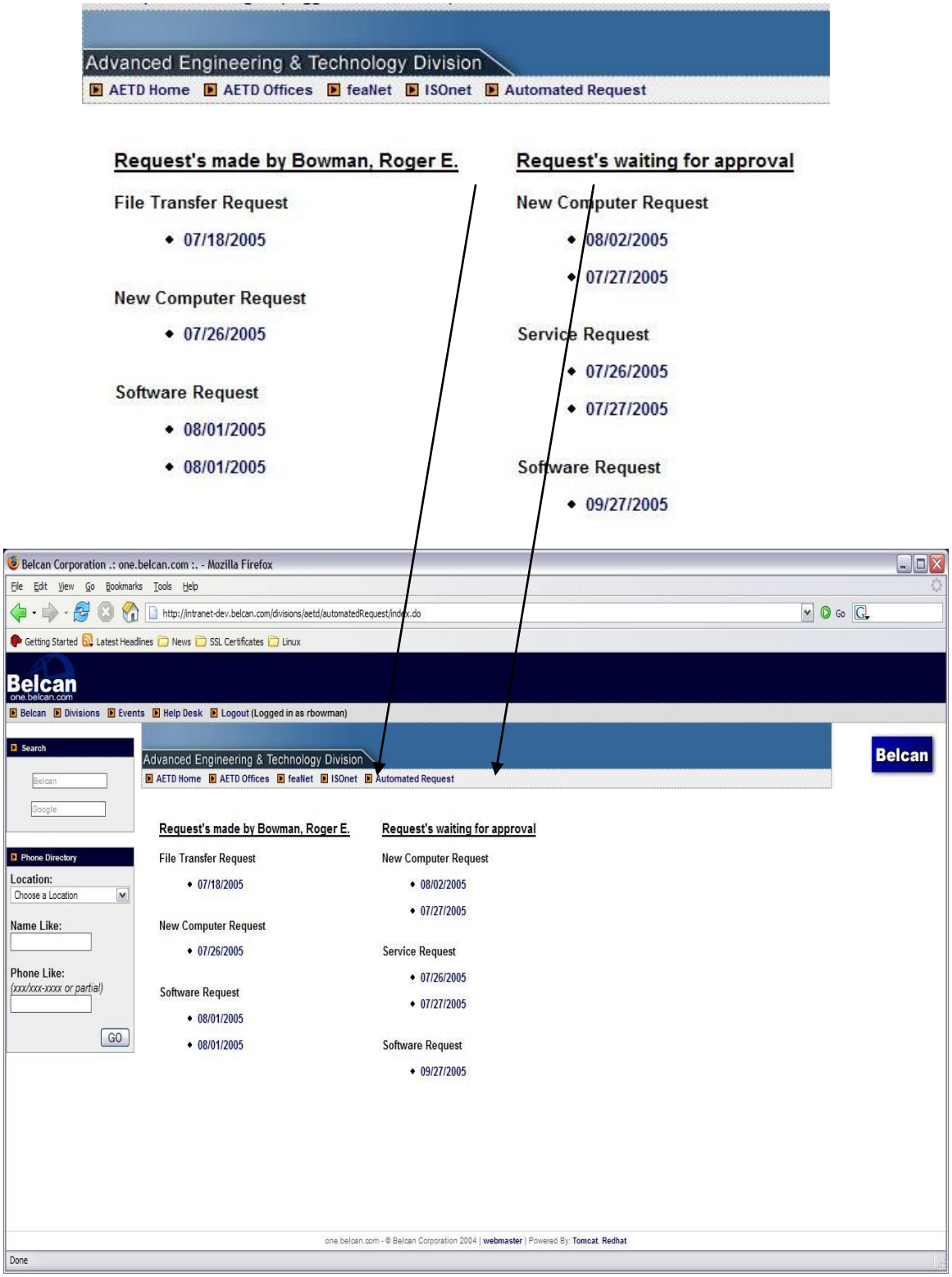


**Figure 4: Manager's home page**

### 2.5.3 Technicians Interface

The technician's interface is similar to both the employee's and the manager's interfaces, but with the difference that this interface allows for the technician to view all of the requests that are waiting for implementation. In Figure 5, it is shown exactly what a technician will see upon logging into this section of the site.

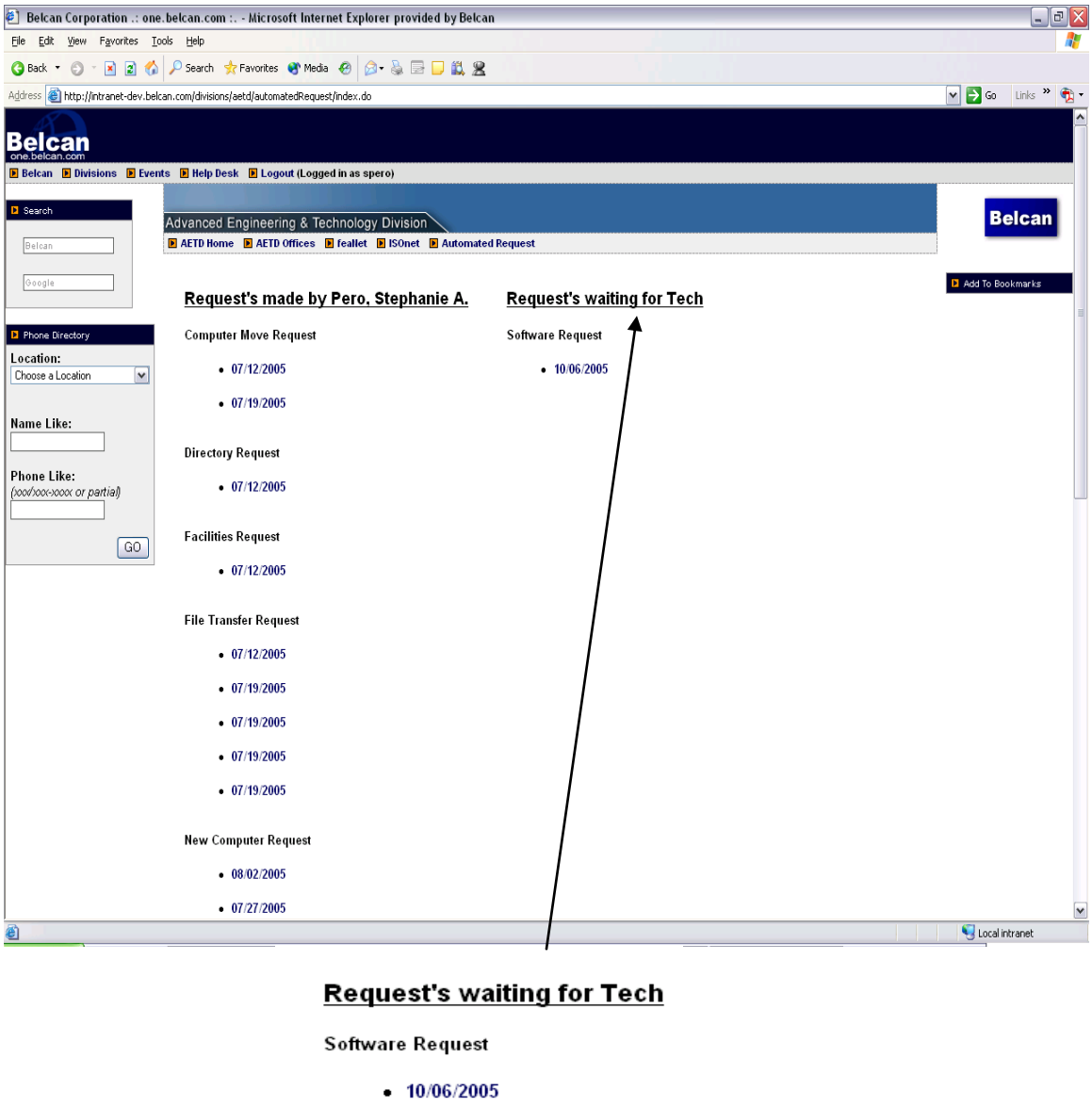





**Figure 5: Technician's Interface**

### 2.6 Database Design

In the creation of this application a backend data repository was needed in order for this application to work as intended. I chose to use PostgreSQL for my database management system (DBMS) for several reasons which include; all of Belcan Corporation's current web applications and web sites run with this particular DBMS, in the requirements of this project it was clearly stated that this system must parallel with existing forms, processes, and work instructions, it also stated that I use currently owned products. Along with some of those reasons PostgreSQL is a true relational database that allows for primary and foreign keys to be created easily.

In the creation of the database I found that I really only needed 7 tables to have a fully functional and normalized system. Each of the four forms has its own table that holds all the pertinent data for each request. The other three tables consist of the technician's form, the employee's manager's list, and finally a status table. These forms work together to form the backend repository of the application. Figure 6 is a database diagram which illustrates my chosen design.

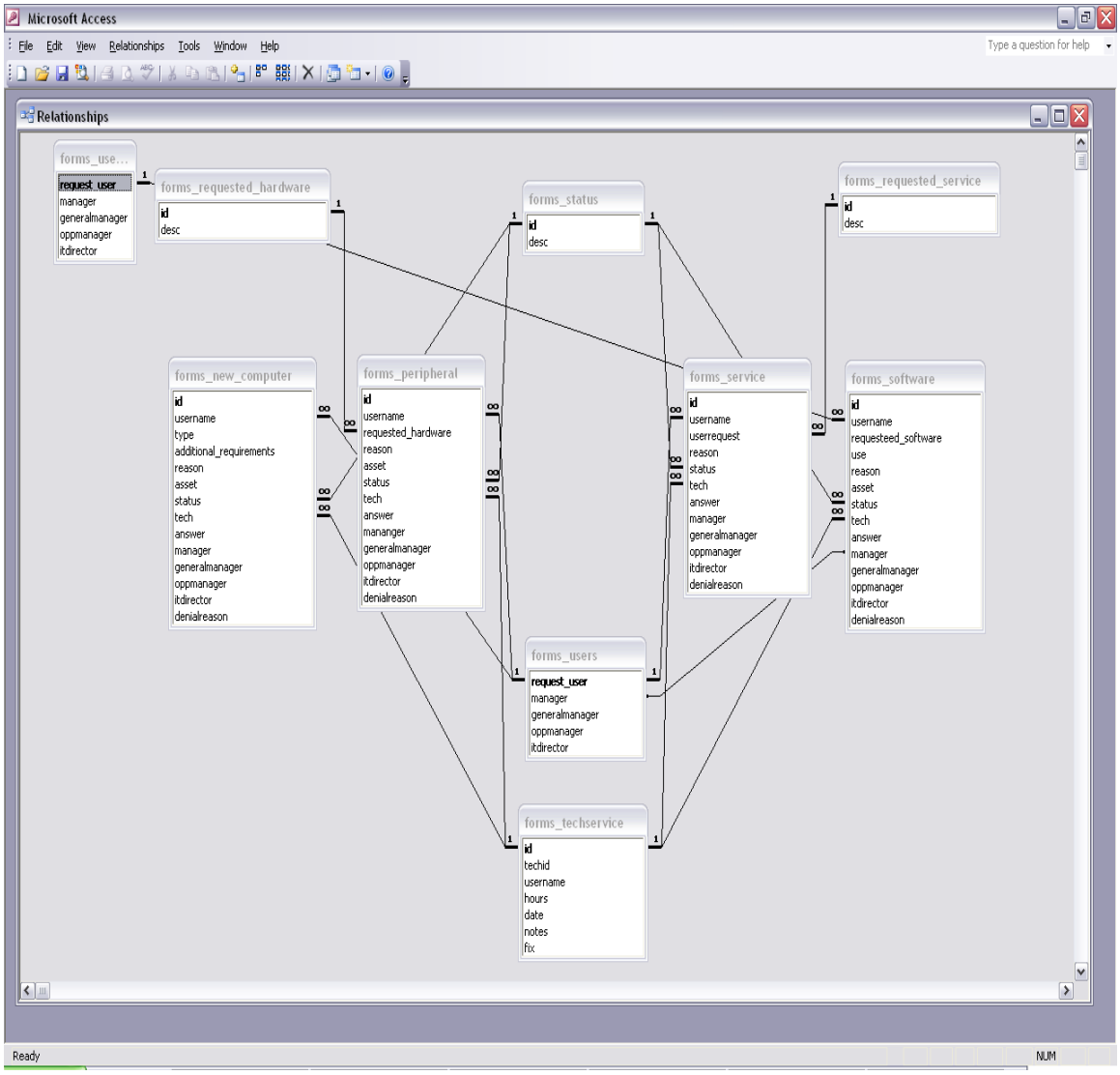


**Figure 6: Database Diagram**

### 2.7 Search Engine

Initially when designing the Automated Request System a search engine was intended to be utilized for the technicians to search through all of the completed computer related requests. Once the system was put into place and the majority of the development had been completed I realized that the third party search engine was not compatible with my system. The problem became apparent when some security features were added in to the entire system. With the added security, the search engine could no longer effectively index the site and retain the needed information for an effective search. This ultimately made the utility useless when it came to the initial effort to bring this software into place.

### 2.8 Business Logic

The Automated Request and Document Management System has a business layer that is made up of all of the class files that control how the system interacts with the input data. Basically there are four top level sections which include the New Computer Request Form, the Peripheral Request Form, Software Request Form, Service Request, and the Technicians Qualification Form. Three of the four top level sections have four subsections which include a form class, submit class, view class, and a form bean. Below, in Figure 7 and 8, I have a high-level overview of a class diagram that illustrates the different classes and top level sections that are involved with the system and a more in depth view of the entire system.

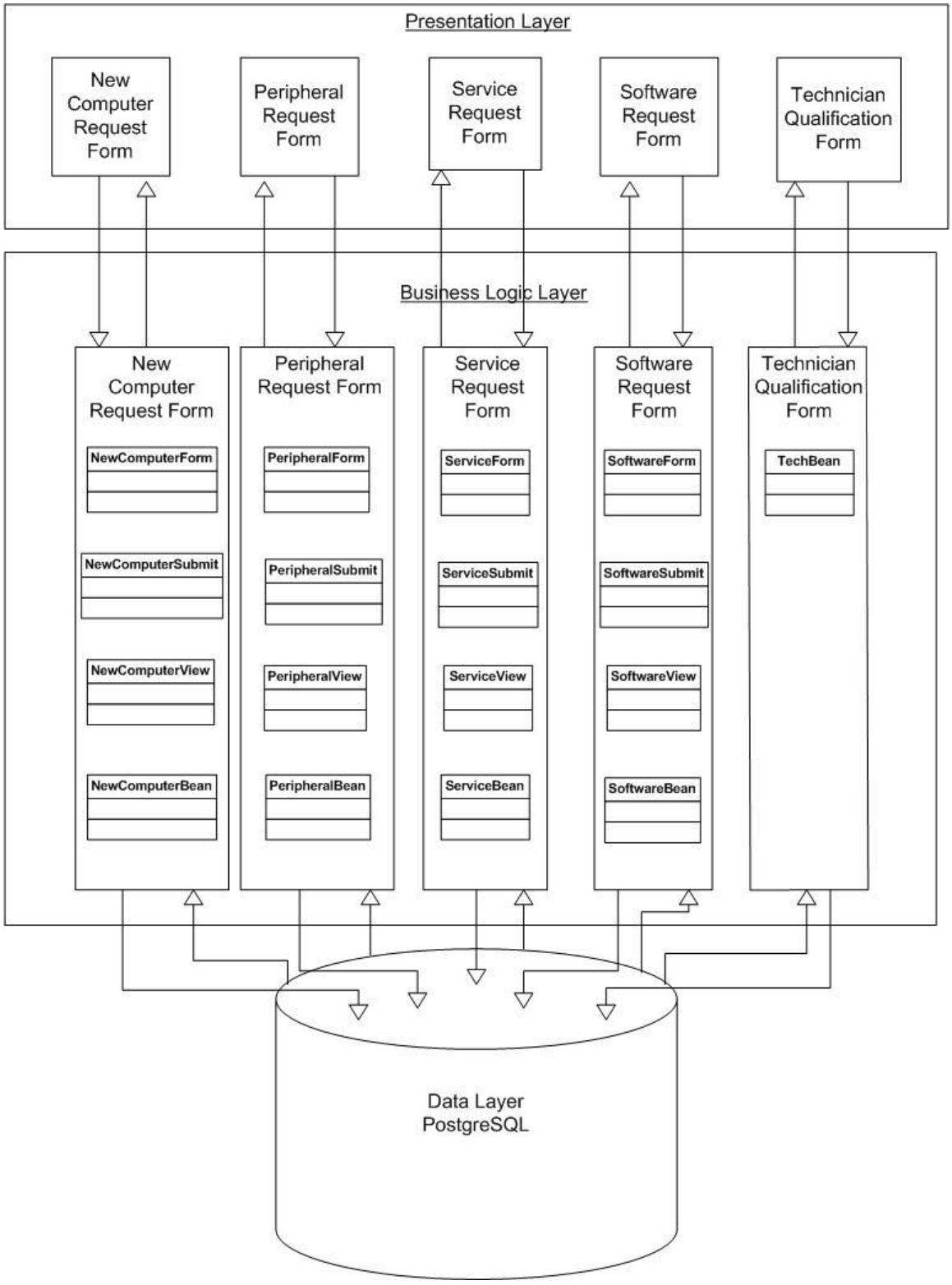


**Figure 7: Complete high-level Overview of System**

Besides these class files there are other components that interact with each other that make this application work and function properly. For example the validation that is done on each of the forms is controlled simply using a struts tag (Apache Struts is a project that was created by the Apache group that is designed to encourage the use of the model2 architecture. The model2 architecture is based on the separation between presentation and content, which simply takes servlets and manages all the business logic, then, uses a separate layer that handles all the presentation through the use of java server pages) in the java server pages. The tag is simply one line of code that looks like this <html:errors/>. This simple line of code is what triggers the validation to take place. All the validation is located in the form bean, but that isn't the all of the code that makes it a working validate function. There is still one last piece that is located in a properties file. This last piece is the actual statements that you will see once a validation error is made on one of the forms. An example of such a function is located in Appendix B.

In these classes there is a certain functionality that makes this system work proficiently and dynamically, and that is the email notification. This simple function is what makes this application such a robust and dynamic system. While the rest of the application plays an important role in the creation of an automated system, the automated email notification is what makes this a truly user friendly automated system. The employees at Belcan no longer have to worry about who these requests go to because they are now all automatically sent to the appropriate people. An example of the code can be found in Appendix B with a full statement explaining the code.

## 3. Deliverables

For this Web site to be a robust yet completely user friendly site there were certain items that were needed to ensure success.

- A Web Based Automated Request and Document Management Systems that dynamically notifies user, managers, techs, and upper management about requests.
- The user interface is written with java server pages, and it also uses java beans, and java classes. This allows for robust but easily navigated user interface.
- Users must log into the site to submit or view requests.
- The site connects to Active Directory for fast retrieval of user information and pre-population of certain fields, this is why it is important for user to log in.
- The user will be able to do the following.
    - Add Request
    - Review Request
    - Edit Request
    - Submit the Request
    - Receive E-mail Notification about requests
    - Check on status of request
- The managers, general managers, operations manager, and IT director will be able to do the following to a user's submission.
    - Approve or Deny Requests
    - Submit Requests
- The Technicians are able to do the following:
    - Implement the solution

- Input notes about request
- Close out the request for final notification of completion.
- Search through past request

## 4. Testing Plan

During the Implementation of this project there are many different types of testing that have been done and that still need to be done. There are some key questions that I need to ask myself when deciding what types of testing that I am going to use. The three that are most important right now are high level questions that should be known early in the heavy testing phase. The first question is; what units will I be testing? I need to have a full understanding as to what I am actually going to be testing before I can start. Also, who is going to do the testing? I need to figure out if I am going to allow actual users on the system or if I am going to keep it confined to just the computer services group. Last question is; when will I be testing this? For this last section I need to decide when the best time to do each of the different testing phases is.

### 4.1 What

In the current system there are a few components that really make up the bulk of the system. The different units that will be rigorously tested will include the requesting and submittal of an item, the approval/denial process, and the technicians adding notes and finally closing out the item, search capabilities, and all of the notifications that come along with each section of the site. Each of these sections will be tested using unit testing done by a small group of users so that as many input/output combinations can be tested.

### 4.2 Who

Right now the only people that will have testing access to this system will be computer services group members and the web applications development team. They will use JMeter, which is a load testing utility that is made by Apache. With some time users may be asked to do some light testing just to see what they might input into the system. This will assure that the development and testing done by myself and other IT staff will be as close to what users will put as possible.

### 4.3 When

This testing will be an ongoing ordeal throughout the remainder of the testing phase. I have already started unit testing and will continue also all of the load testing has been started simultaneously to ensure that no matter what we throw at the server it will be able to handle it.

## 5. Design and Development

In the next two sections the timeline and budget are explained thoroughly.

### 5.1 Timeline

| **Senior Design I** | **Complete** |
|---|---|
| Research | Yes |
| Create Forms | Yes |
| Create Database | Yes |
| Create user interface | Yes |
| Create connection to DB | Yes |
| Test user interface | Yes |
| Test connection from interface to DB | Yes |
| Create Validation | Yes |
| Test that Validation works properly | Yes |
| Create automation for Documentation | Yes |
| Test Documentation functionality | Yes |
| Create Notification of Status | Yes |
| Create E-mail Notification | Yes |
| Test Status and E-mail Notifications | Yes |

| **Senior Design II & III** | **Complete** |
|---|---|
| Load Test System | Yes |
| Test all individual parts of system | Yes |
| Make design of application more appealing | Yes |
| Comment all code with JavaDocs | Yes |
| Final Testing | Yes |
| Make sure everything is in order for Final Presentations | Yes |

**Figure 8: Time Line of Project**

### 5.2 Budget

The budget for this project is displayed in the Figure below. With this project taking place for Belcan Corporation everything that was needed to make this into a reality was provided at no extra cost. I did take the time to ensure that the information I provided was accurate so I did some research to find the approximate prices for each item if it were not already owned.

| | Product | Cost | Belcan's Cost |
|---|---|---|---|
| **Hardware** | PowerEdge 2650 | ~$8,000 | Already Own<br>$0.00 |
| | PowerEdge 750 | ~$1,400 | Already Own<br>$0.00 |
| **Software** | Apache 2.0.50 | Open Source | $0.00 |
| | Tomcat 5.0.28 | Open Source | $0.00 |
| | postgresql 7.4.6 | Open Source | $0.00 |
| | SuSE Linux Enterprise Server (SLES) 9 | $35 – Media<br>$899– per server/per year | Already Own<br>$0.00 |
| | Adobe Photoshop 7.0 (16) | $169.00 | Already Own<br>$0.00 |
| | Macromedia Dreamweaver MX 2004 | $239.99 | Already Own<br>$0.00 |
| | Eclipse 3.0 | Open Source | $0.00 |

**Figure 9: Budget**

## 6. Proof of Design

This section it will be explained how each of the deliverables was met, and if any problems occurred in the development and implementation of the project.

This system is made up of many components that needed to be developed, implemented and made into one in order for this system to become what it is now. At a high level viewing of this application it is made up of 3 tiers which consist of a presentation tier, business logic tier, and a data repository tier. These tiers work together to allow for this system to function as a whole and to appear seamless to Belcan employees. While creating each of these tiers I found that it was easier to build components for each tier as I went along, rather than creating all the different components separately and then trying to integrate them into one system. Once each of the components was mostly completed I then added all of the separate functionality such as email notification. This posed little problem because I was able to write one class file that would in turn be used for each of the different notifications.

### 6.1 Example of New Computer Request Process

The presentation tier is made up of user interfaces that were programmed using java server pages that interact with quite a few other components. Some of these components include Active Directory, Open LDAP, PostgreSQL, Java Classes, and Java Beans. These components all are a part of the business and data repository tier of this system and act in different ways to make this application seamless to Belcan employees.

When an employee gets on to the Intranet they are directed to the Belcan Home page as shown here in Figure 10. Employee's have the ability to login from here via the link located in the main header.

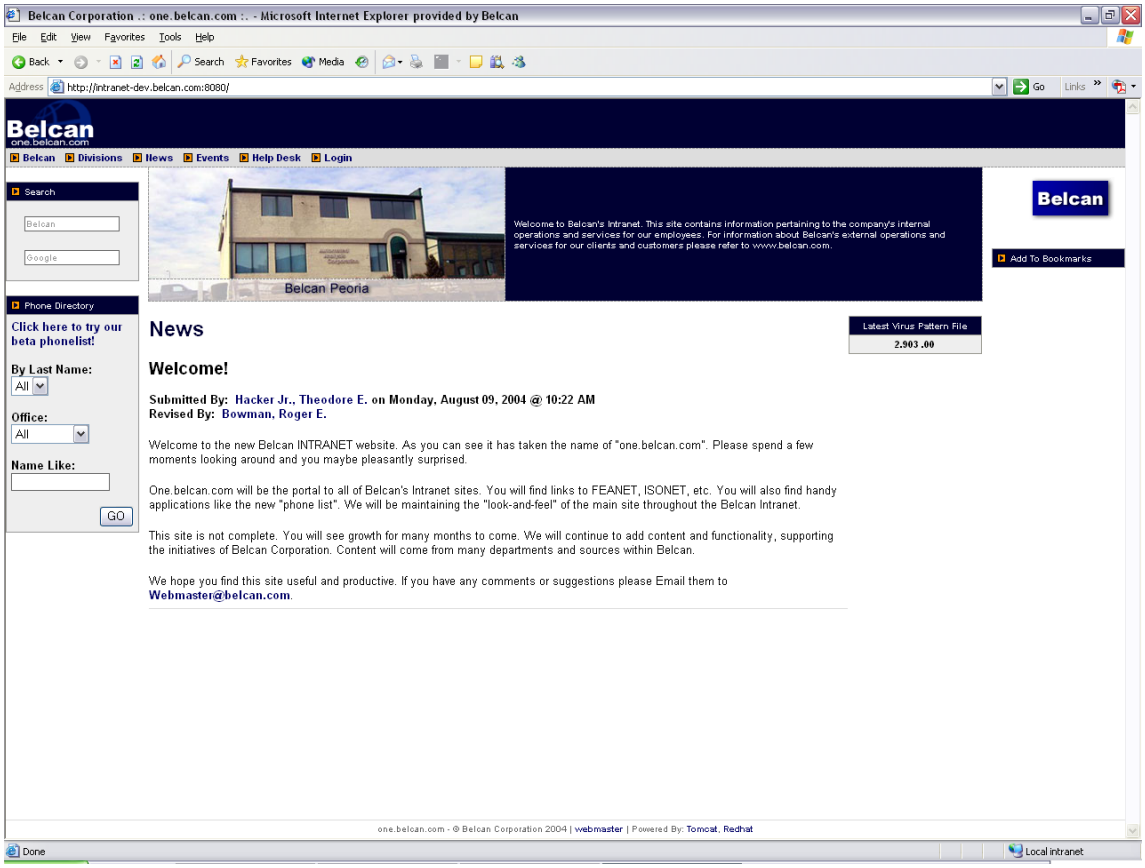


**Figure 10: Belcan Corporation's Intranet Home Page**

Each user must log into the application in order to view any of the forms or to view their own home page. This login makes sure that all the information coming from the request is genuine information. It also ensures that the correct managers and individuals receive the correct information.

The Automated Request Process and the Document Management Systems both use Active Directory to authenticate users and store information about the individuals. Without the system connecting to Active Directory each user would have to manually input basic information such as user name, office location, division and so forth. This information is all retrieved on the spot when the user logs into the system. This stops

some of the human error that is involved when we are asked to type in a lot of information. It also ensures for the quick lookups of users and their information.

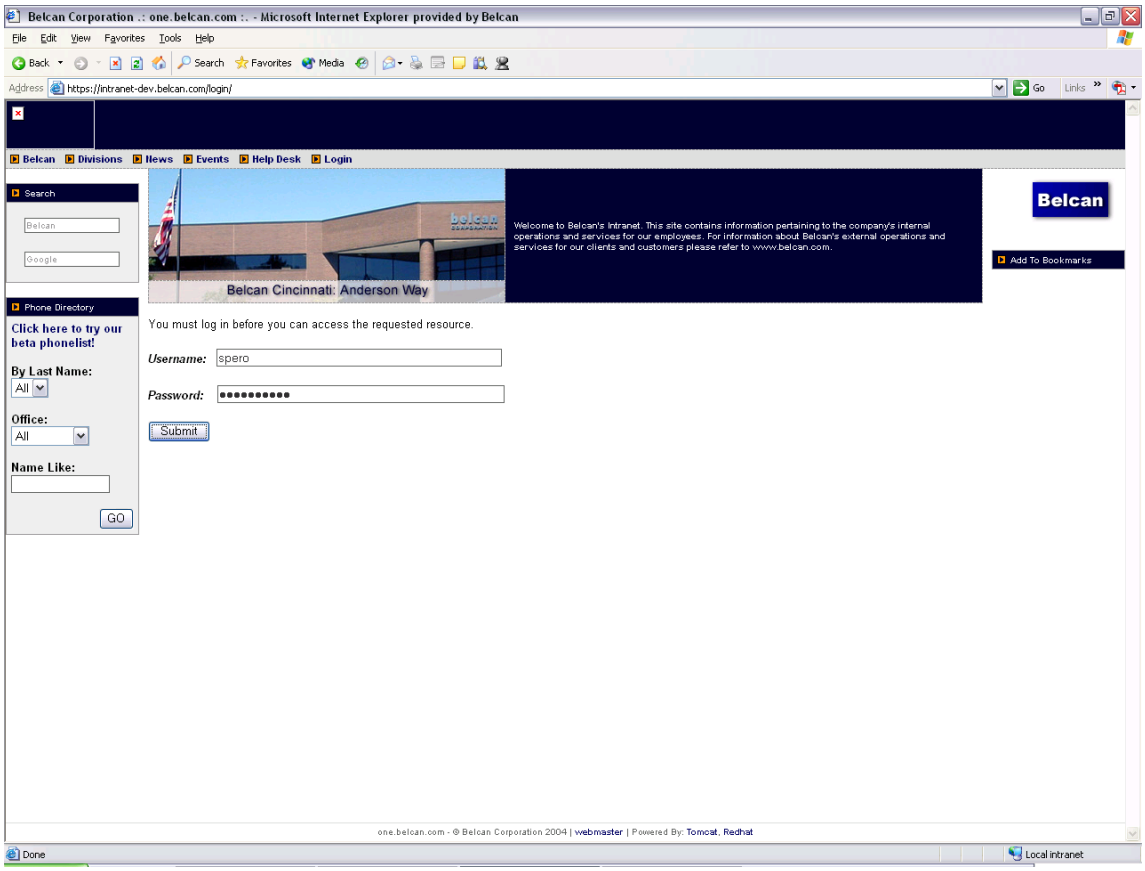


**Figure 11: Login Screen**

Once the user is logged in they have access to all of the automated request forms. They choose which form they need to fill out and click on that link. This is displayed below in Figure 12; the employee is selecting the New Computer Request Form. The employee is then brought directly to the empty form.

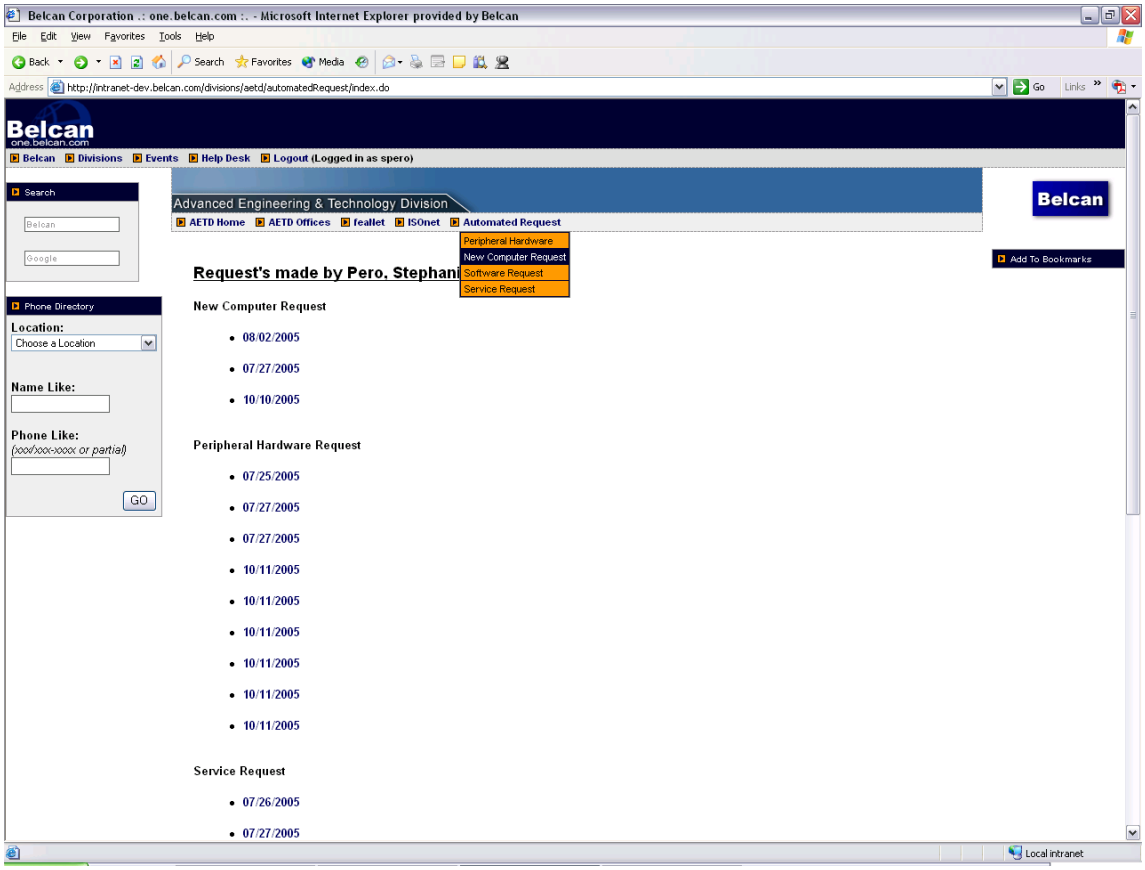


**Figure 12: User Selecting Automated Request Form**

The employee then fills out the form, as shown below in Figure 13, with valid information and submits the form for review. Once the employee has reviewed the request and everything is how they want it, they then submit the request and it is sent off to their direct manager.

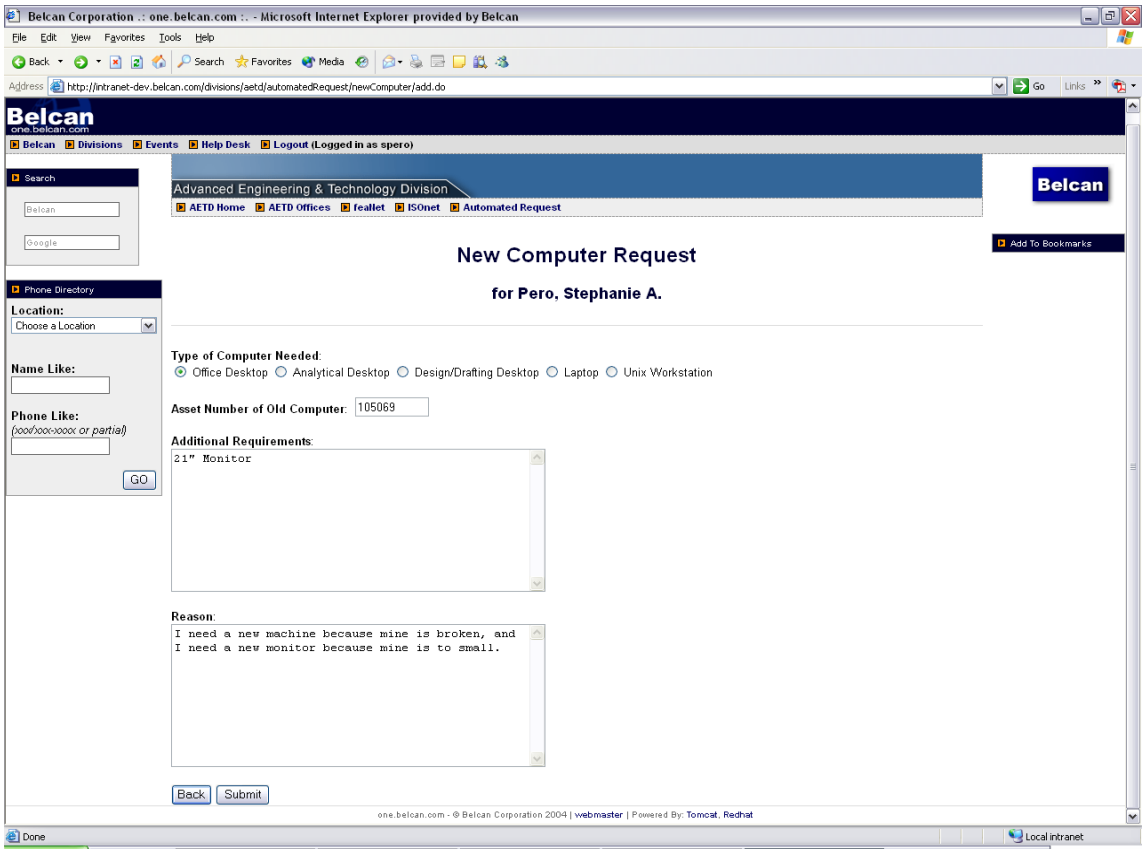


**Figure 13: Completely Filled Out Form**

Once the user has successfully filled out the form and checked to make sure everything is correct he/she will then finally submit the form. This will then in turn start the notifications of E-mails to the managers. This process is started by first sending an email to the direct manager. If the direct manager approves the request it is then sent off to the general manager. If for some reason the direct manager denies the request, then the process is stopped there. When the general manager receives the request the same processes takes place and if it is approved it is then sent to the operations manager of that office. When the operations manager has approved of the request it is then sent off to the IT Director for approval. If the IT Director happens to approve the request it is then sent back to the operations manager so that the correct technician can be chosen. Once the

operations manager has received the request for the second time he/she then choose what technician will be responsible for the implementation of the request. Finally, when the tech is chosen, a notification is then sent to the tech for qualification.

The submittal of a form is the "kick off" of the automated process. This is run using a Java Class that has different parameters depending on what form is being submitted.

Once the request is submitted, the employee is then redirected to their home page within the application as shown below in Figure 14.

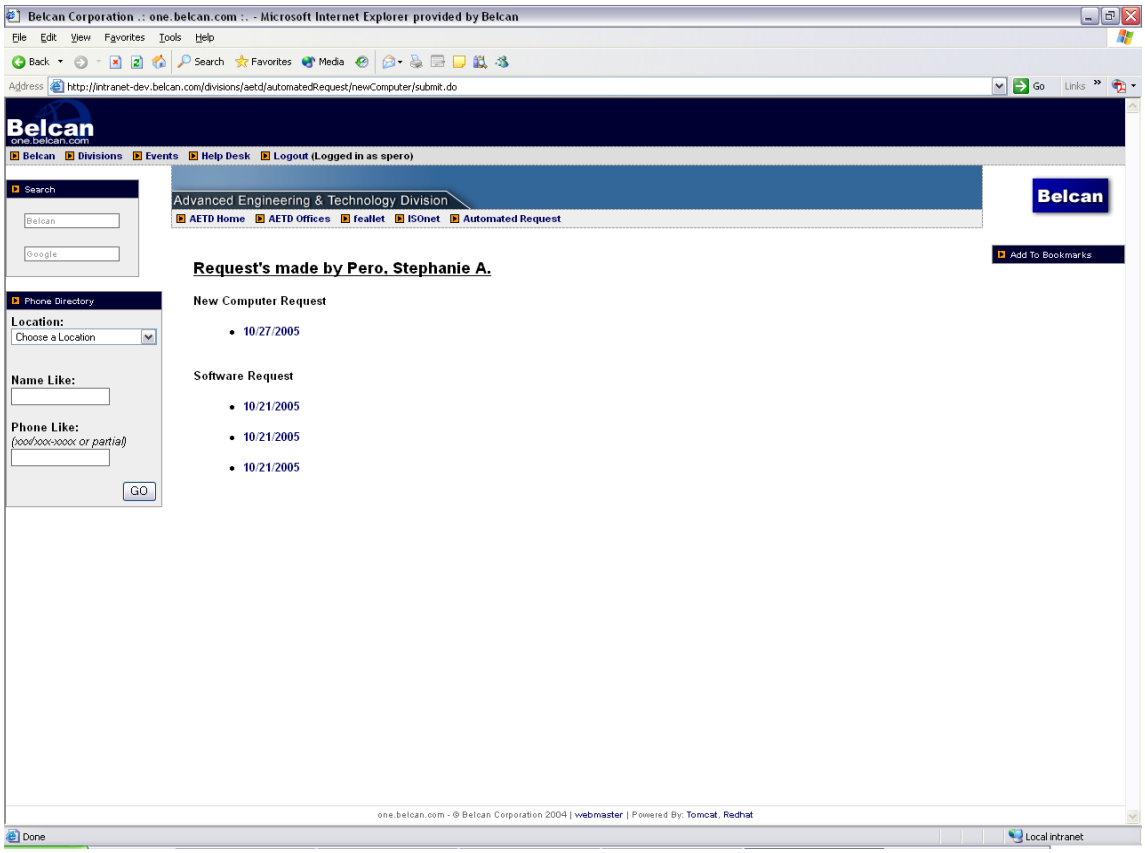


**Figure 14: Employee's Home Page in Application**

When the user submitted the request an email was sent to the direct manager of that particular employee. An example of such an email is shown below in Figure 15. This example explicitly states the employee who sent in the request, and that it is intended for that person's direct manager. The email also has present a link for the manager to follow that will bring them to the approval/denial page.

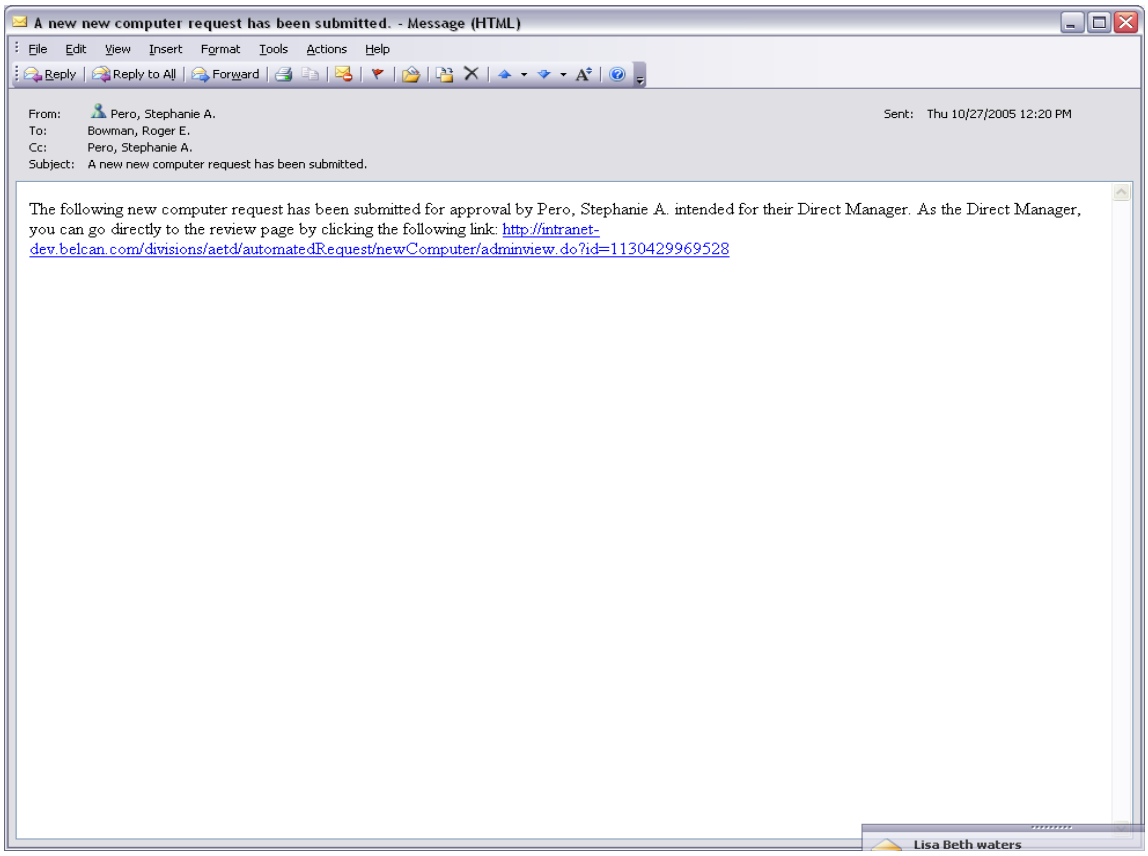


**Figure 15: Direct Manager's Email**

When a Manager logs into the system he/she too have a “home” page. This page is just like that of the user’s home page, but the managers see all of the requests that need to be approved by their direct employees, or if he/she are a General Manager, Operations Manager, or even the IT Director he/she have the same type of home page.

This page queries the database and finds any requests that have their user name for the manager column. Right now this is all hard coded into the system for this project’s purposes, once this project goes into production at Belcan I will then connect to our AS400 for all of the dynamic retrieval of manager’s names.

If the direct manager is not currently logged into the system they will be automatically redirected to the login screen, as shown here in Figure 16.

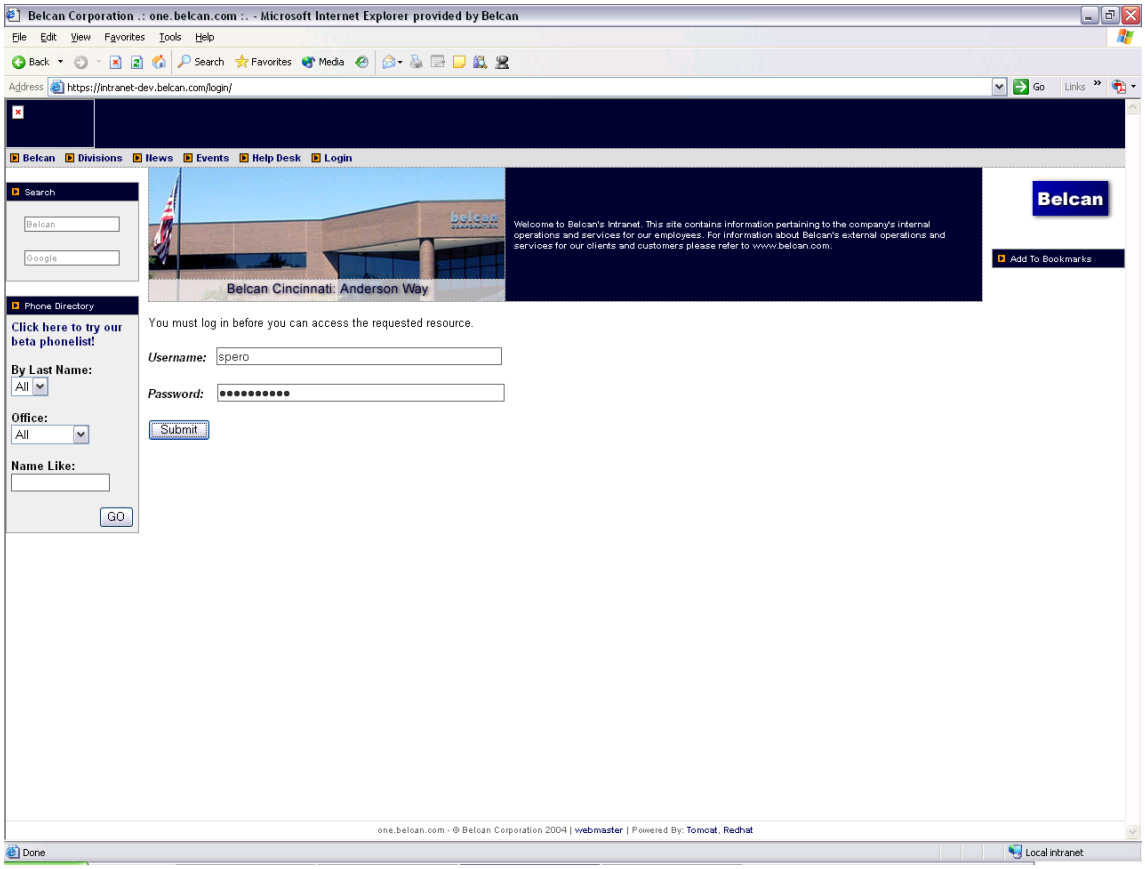


**Figure 16: Login Screen**

Below in Figure 17, is an example of the manager's approval/denial page, when the manager follows the link given in the email they are brought here to review the user's request and either approve or deny the given request. This page allows for each of the different types of managers (direct manager, general manager, operations manager, and the IT director) to approve or deny any request that was intended for them to take care of. Once a manager has looked though the request and decides to approve or deny the request, he/she then click on the correct choice and hit the submit button.

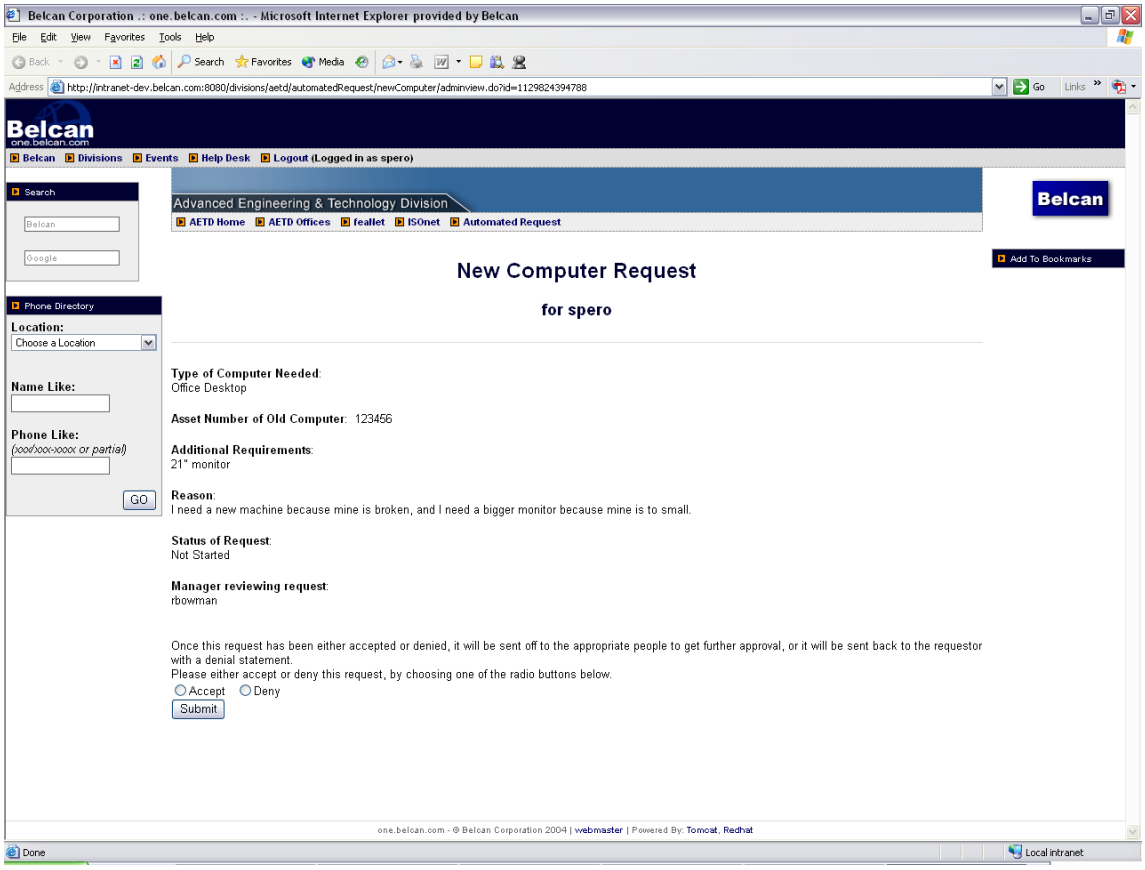


**Figure 17: Direct Manager's Approval/Denial Page**

Below in Figure 18, is another example email that is automatically generated when the direct manager approves a given request. Once the direct manager approves then an email is automatically sent to the initial employee's general manager for further approval of the request. Just like in the previous email intended for the employee's direct manager, the email sent to the general manager also states the employee that the request was initiated by, and explicitly states that this email is intended for that user's general manager.

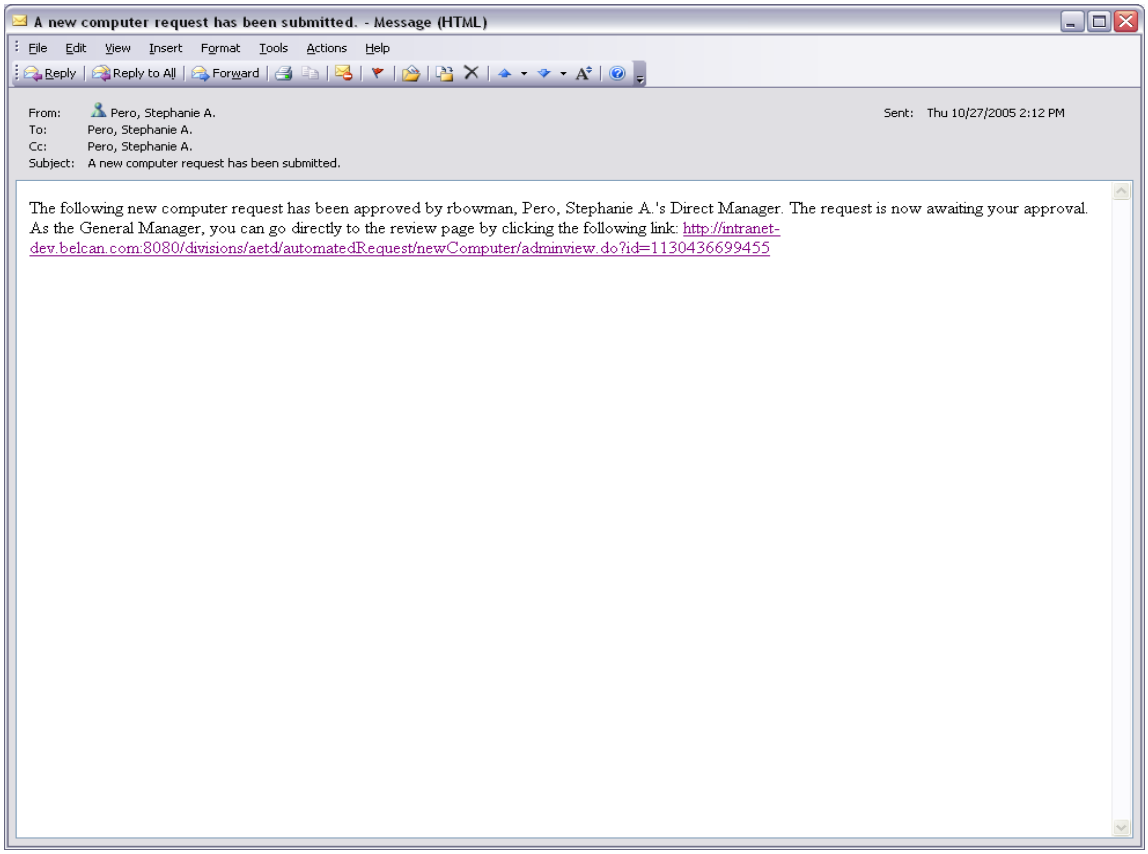


**Figure 18: General Manager's Email**

When the general manager follows the link provided on the email they are brought here (as shown in Figure 19) to the approval/denial page. Once the general manager accepts the request another email is automated and the information is then sent off to the operations manager.

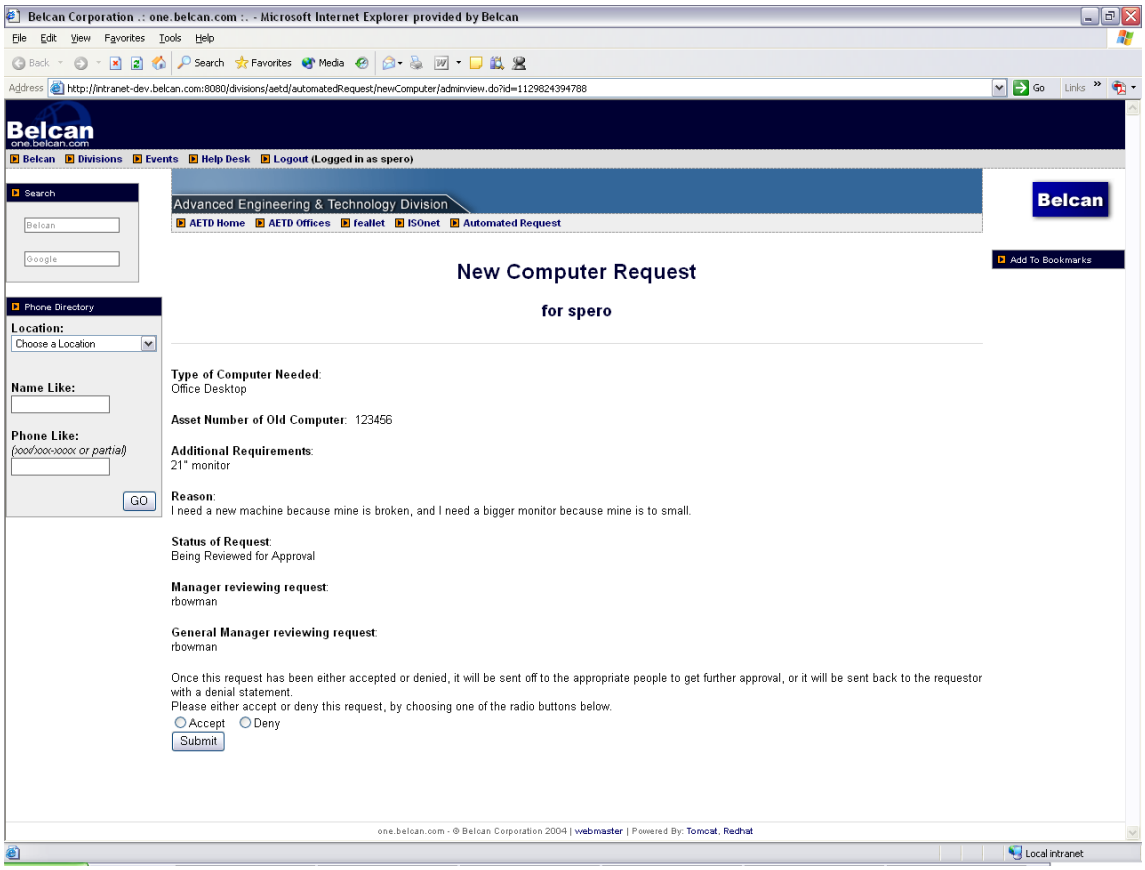


**Figure 19: General Manager's Approval/Denial Page**

Here is an example of the email that is sent to the operations manager. Each of the emails that is automated through this application all contain some important information that shows authenticity such as the username of the approving manager, along with the name of the employee that has initiated the request.

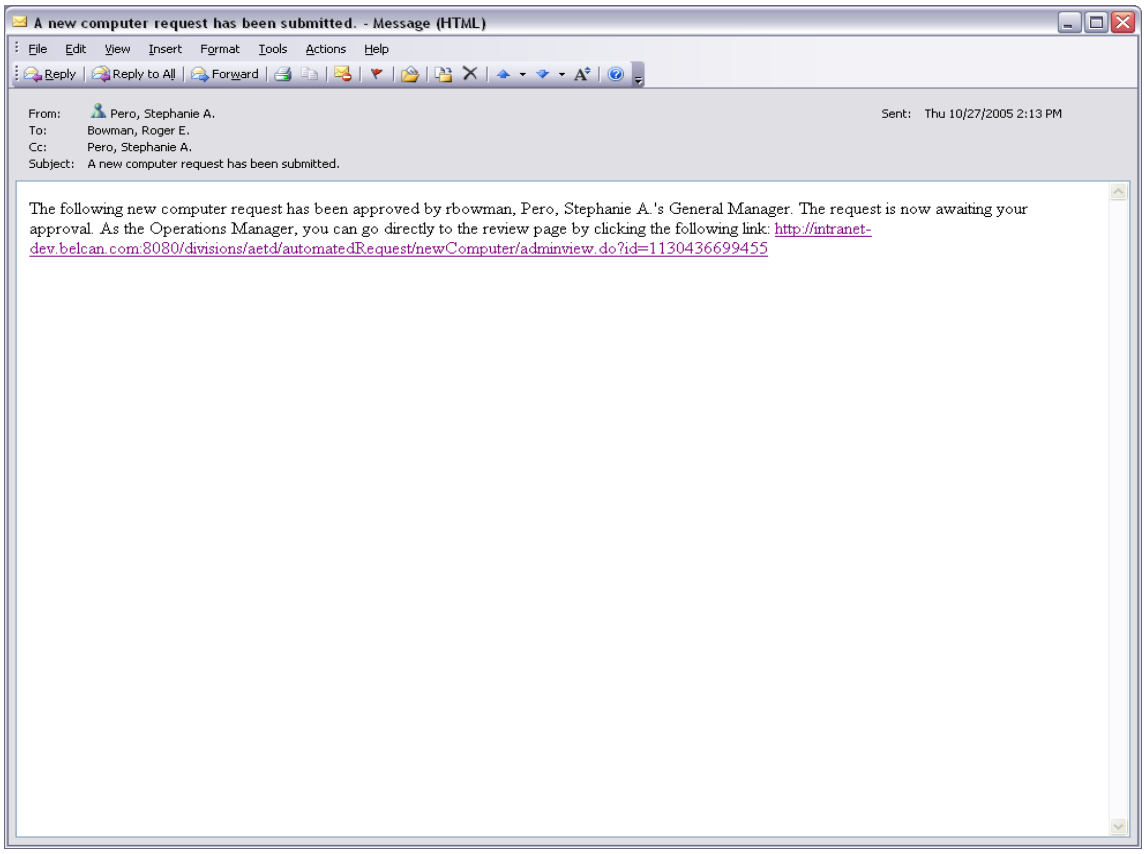


**Figure 20: Operations Manager's Email**

Once the operations manager follows the link provided in the email they are brought to the approval/denial page as shown in Figure 21.

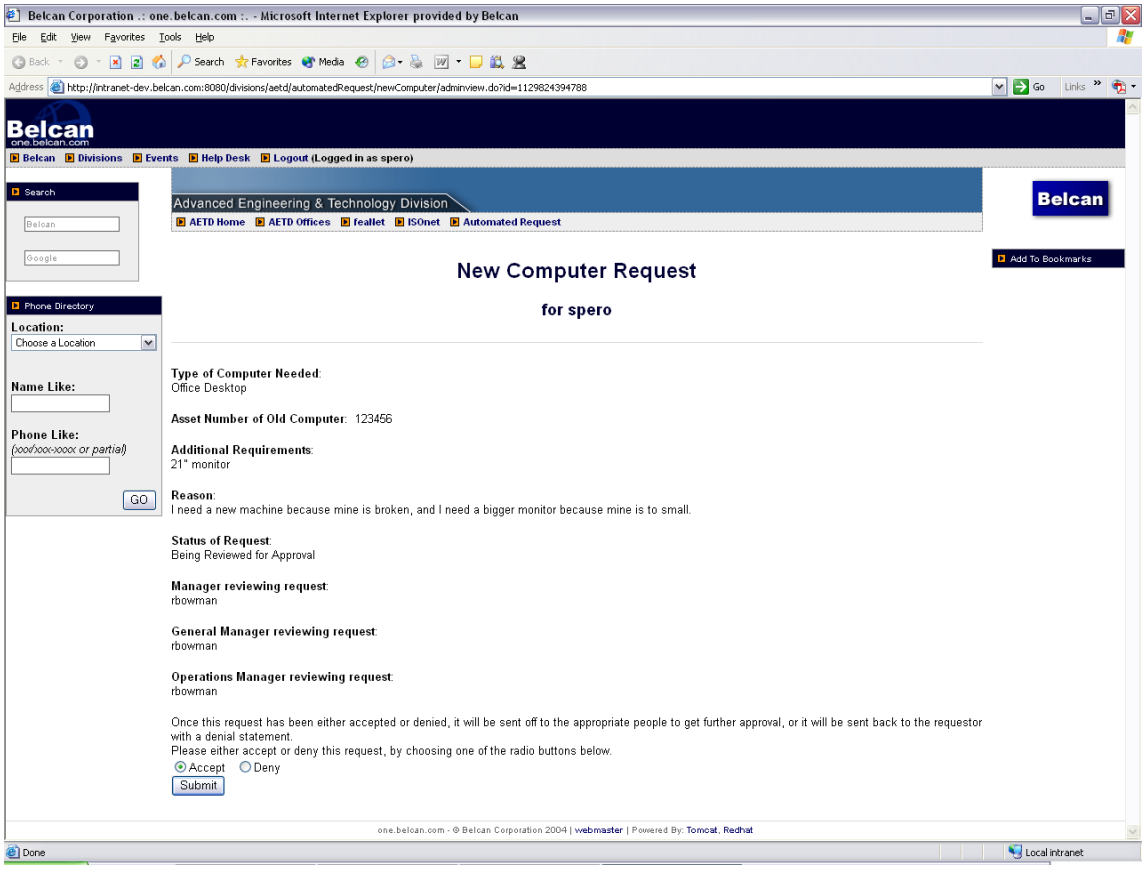


**Figure 21: Operations Manger Approval/Denial Page**

At any point in this process the employee's can check on the status of the request that they submitted. This is done using the "home" page of the user. Each of the requests that are made show up on the home page and are links to the forms themselves, so if a user wants to check the status of a request all he/she would need to do is click on the desired link. Shown below in Figure 22, is an example of an employee checking the status of a request.

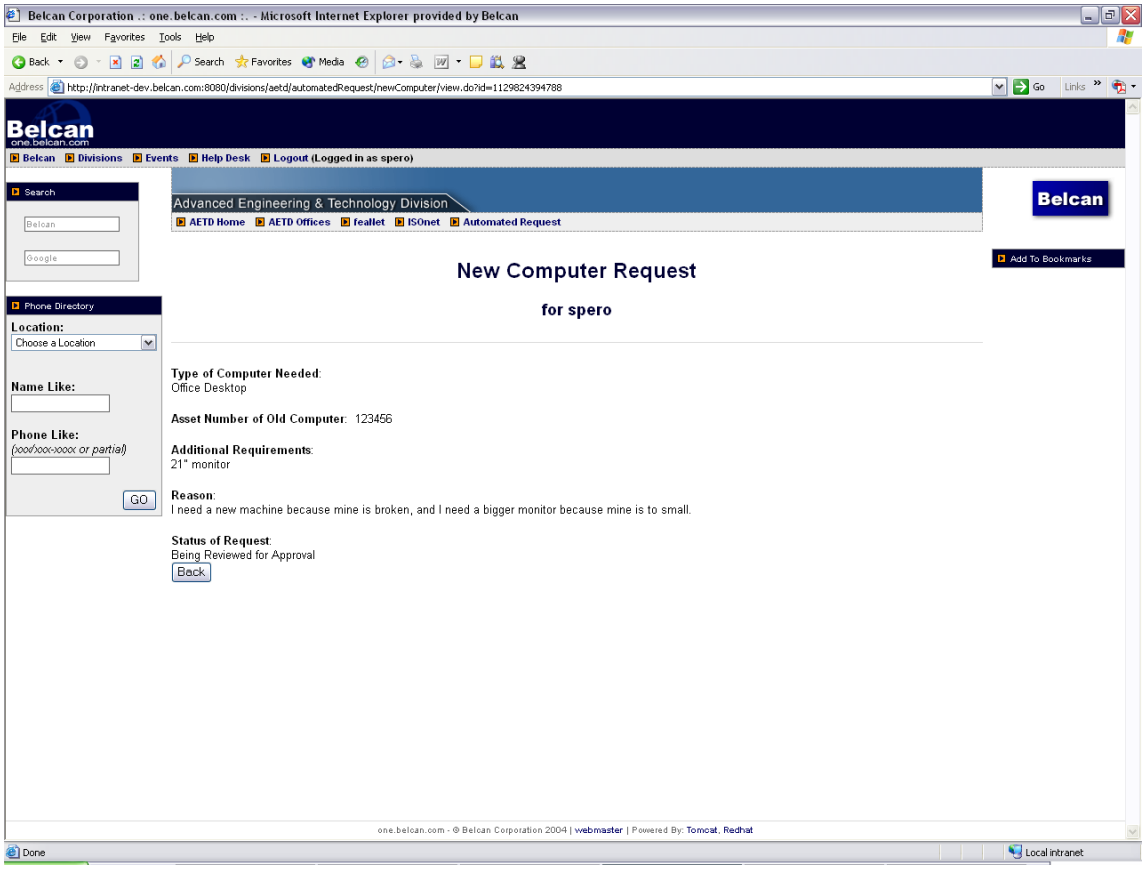


**Figure 22: User Checking Status of Request**

This is another automated email that is sent once the operations manager approves a request. In Figure 23, there is an example of what the IT Director will receive when a request has been approved by all of the preceding managers.

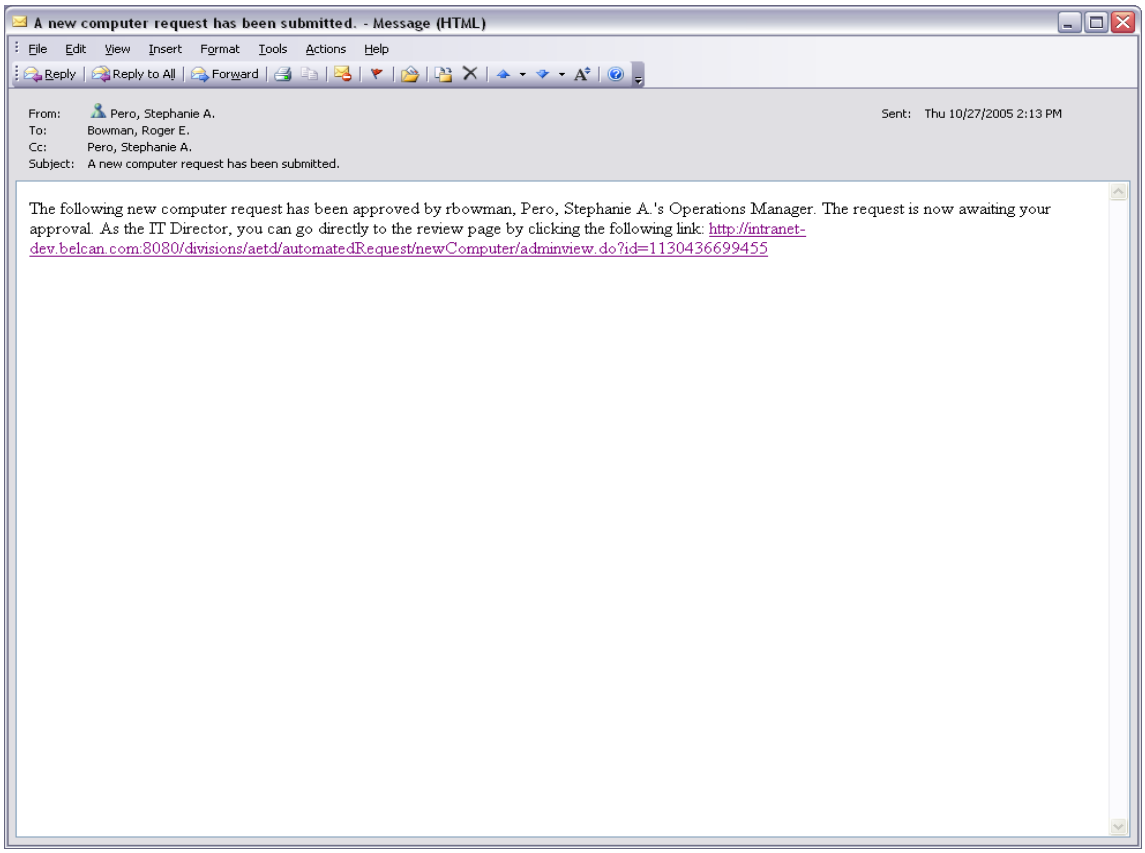


**Figure 23: IT Directors Email**

When the IT Director follows the provided link on the preceding email they will be brought to the following page, shown in Figure 24.

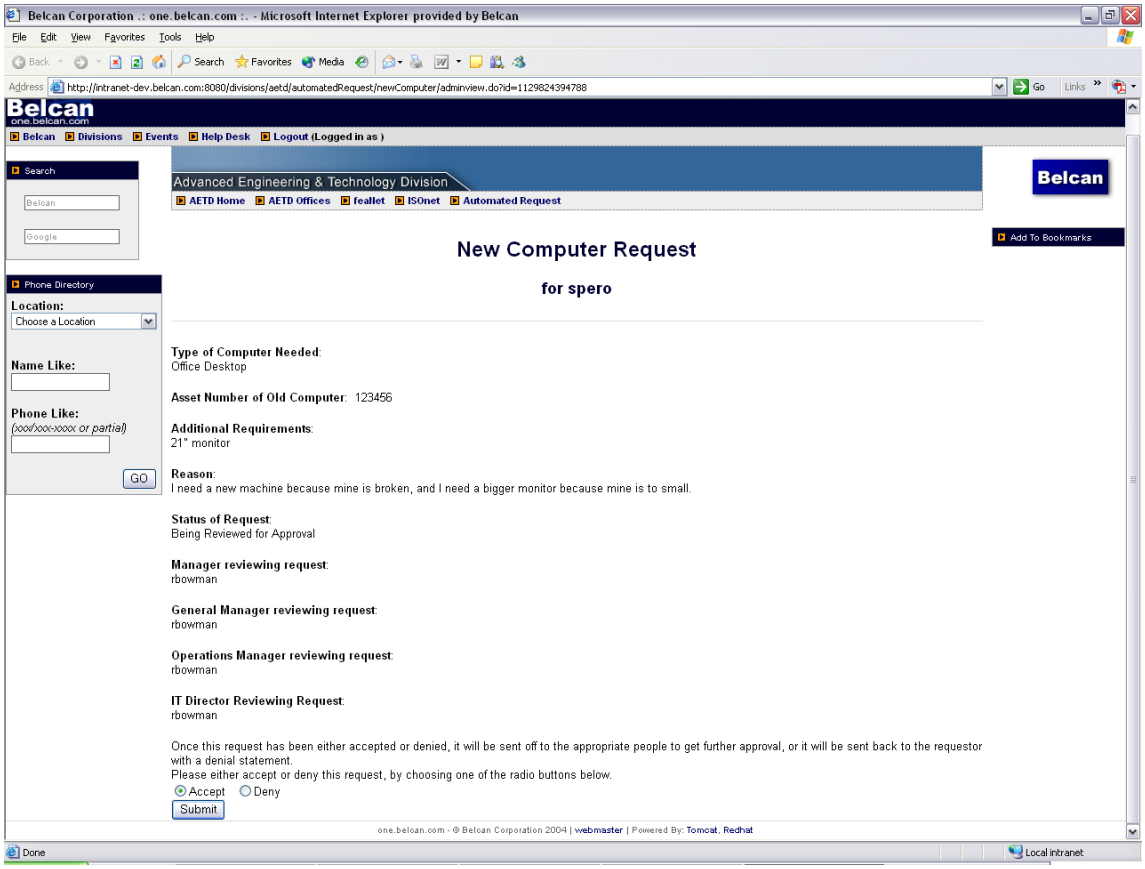


**Figure 24: IT Director's Approving Request**

Figure 25 is an example of the email that is sent to the operations manager once the IT Director has approved the request. This email has a link for the operations manager to follow to choose a technician to implement the request.

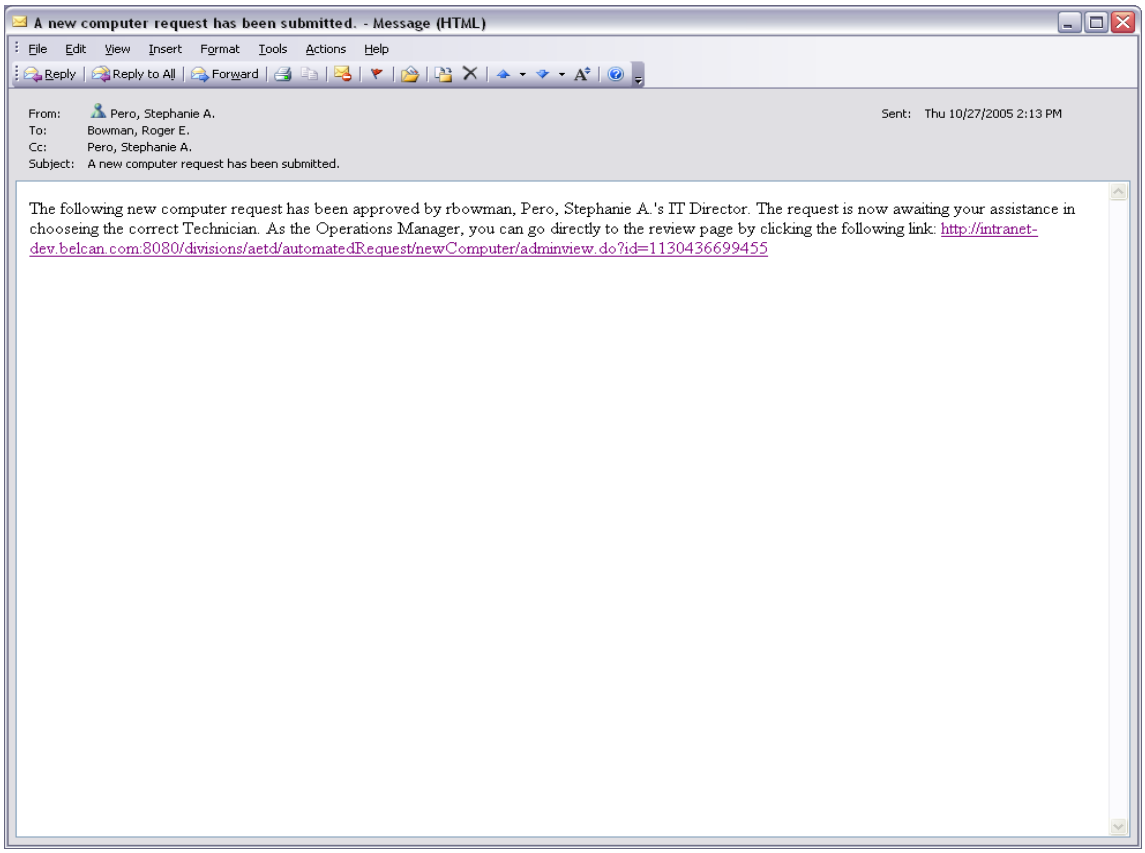


**Figure 25: Operations Manager's email to choose a tech.**

When the operations manager follows the link provided in the email they are brought to the page shown in Figure 26. This is where the operations manager decides who will be responsible for implementing the request. Once a technician is chosen from the list, an email will be automatically generated and sent to the appropriate tech.

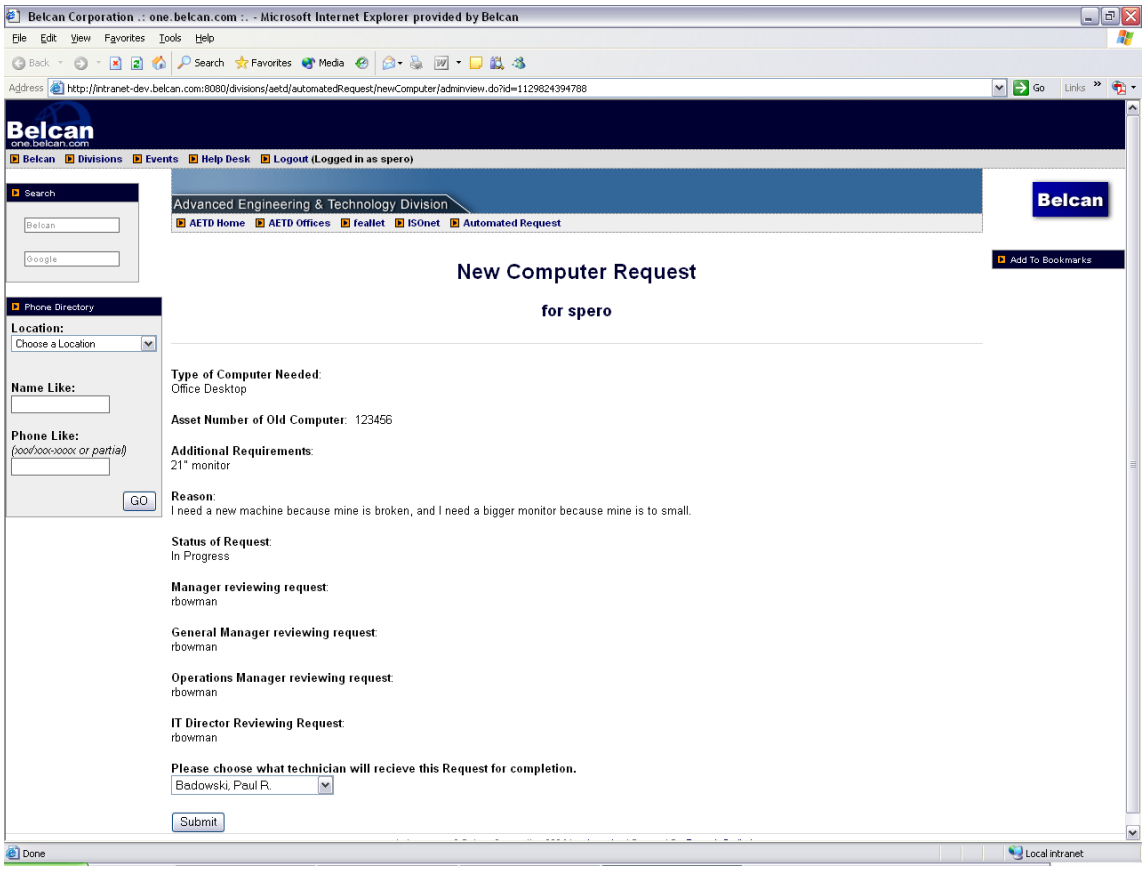


**Figure 26: Operations Manager Choosing Tech**

Figure 27 showing a dynamically populated list of all Computer Services available technicians.

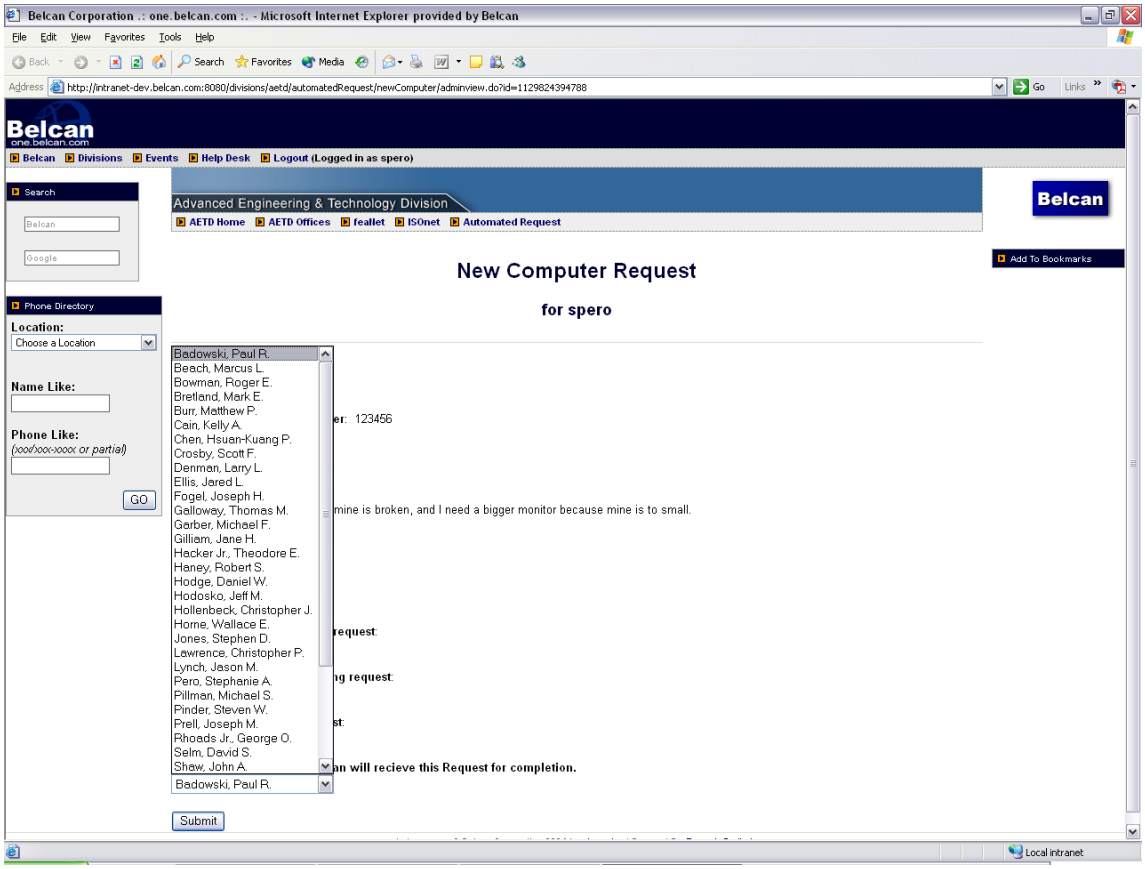


**Figure 27: List of Technicians**

Figure 28 is an example of the email that the technician receives once the operations manager selects someone.

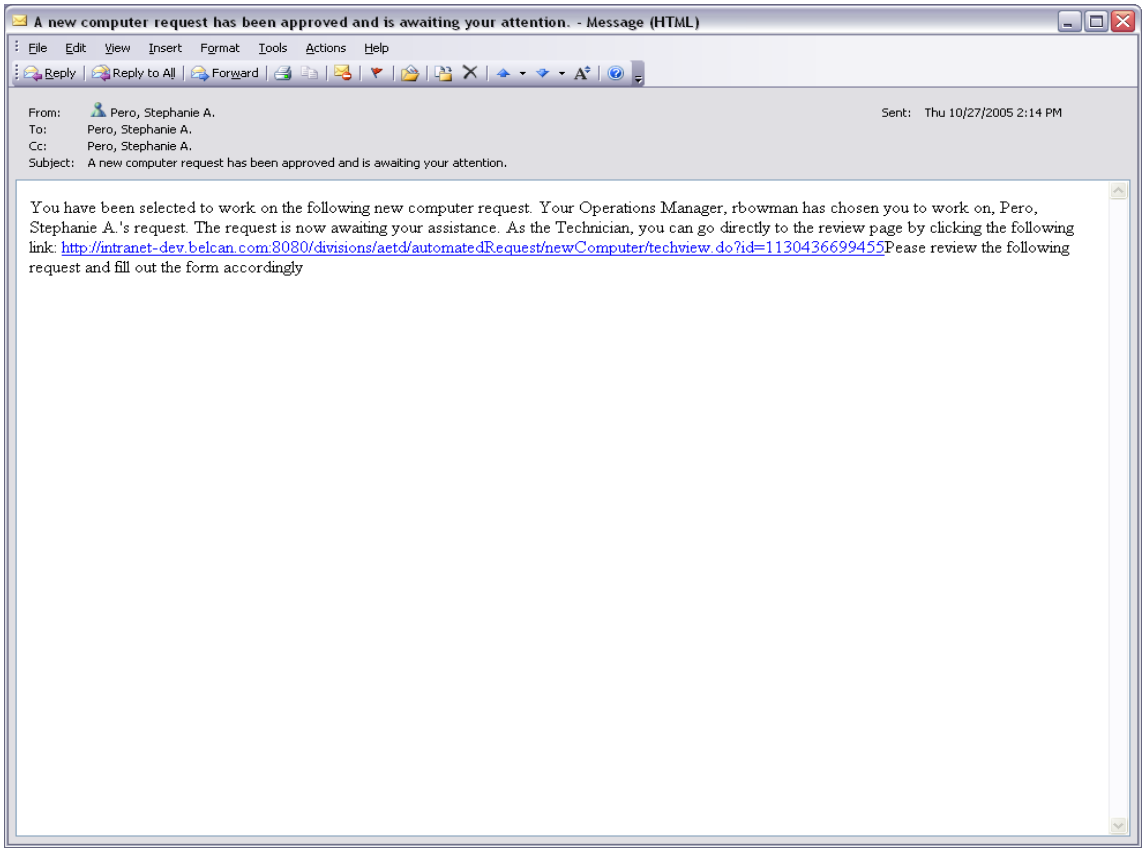


**Figure 28: Technicians Email**

Once the technician receives the email from above they will have the opportunity to follow the link that was provided and it will bring them to this page shown in Figure 29. This image is an example of the technician's qualification form, where the technician will input the approximate hours and date that the request will take to be finished. When the technician chooses an approximate date for the request to be completed they will have the opportunity to choose from a date picker, rather than having to type a date in.

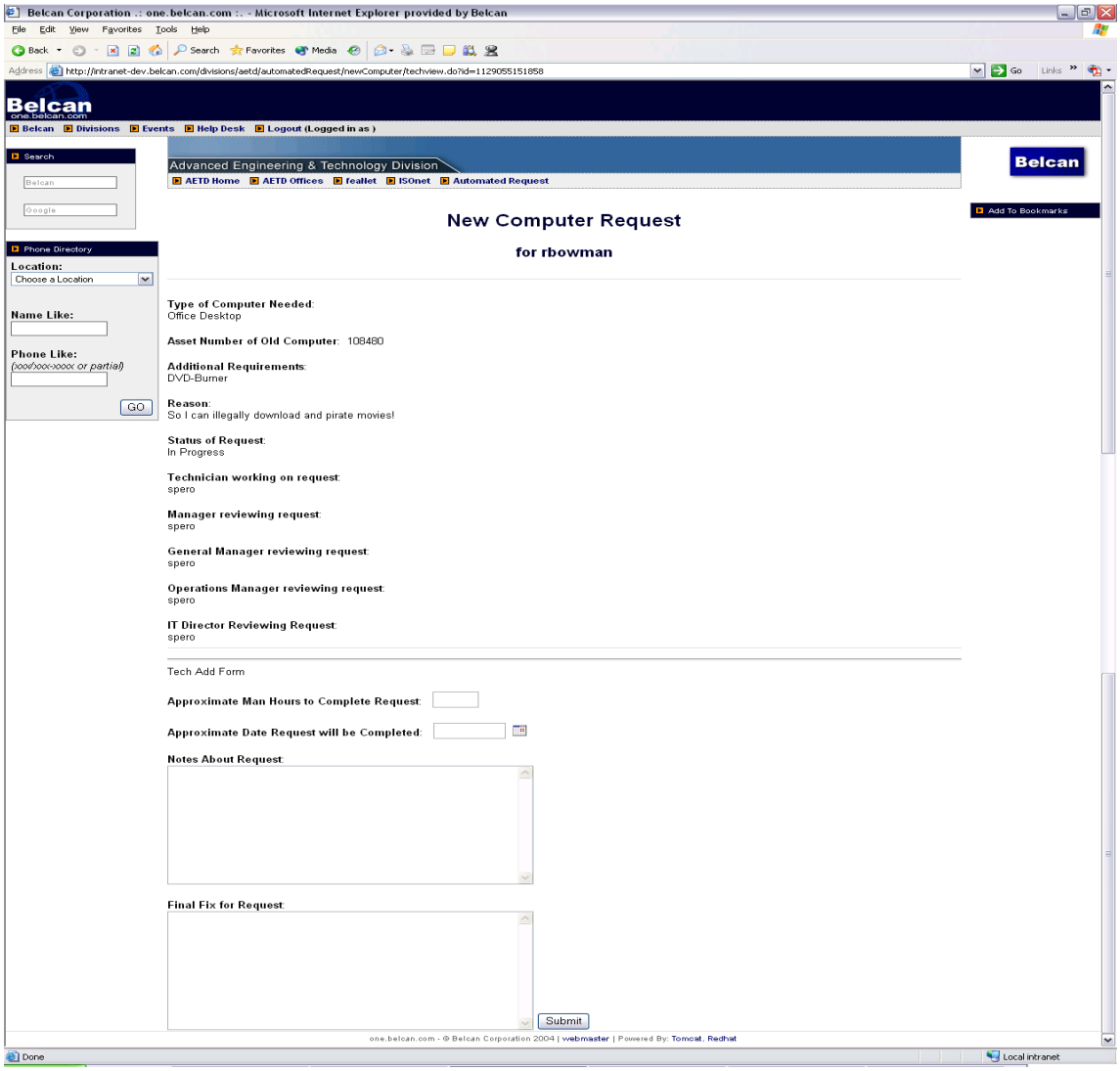


**Figure 29: Technician's Qualification Form**

Figure 30 is an email that is sent to the original employee stating that a technician has reviewed the request.

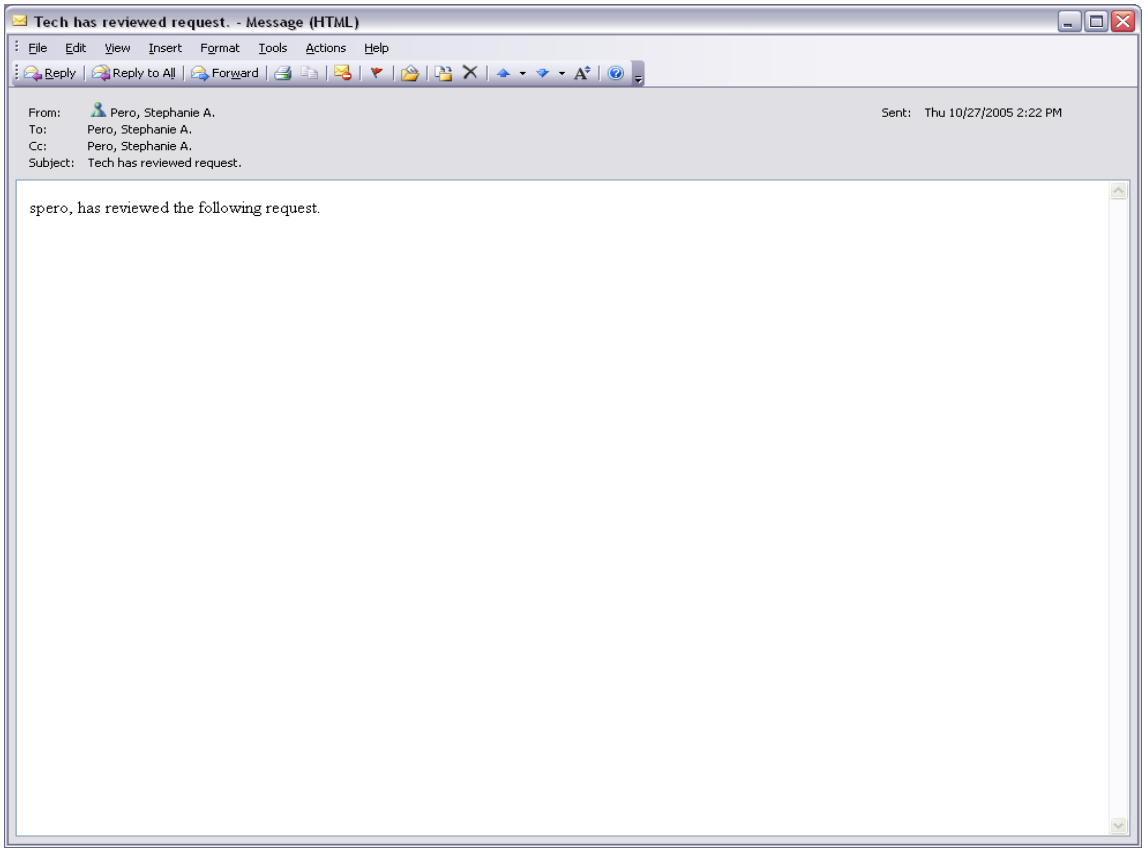


**Figure 30: Email to Employee**

The techs are required to input notes into the form each time he/she work on the request. This makes certain that there is information related to each and every request that has been closed out or worked on. This also allows for each of the techs to go and search the different requests for a future quick fix to a problem that has been resolved in the past.

Once the tech has all of the preliminary work finished it is then time to start working on fixing the problem. Whatever the request may be there is a tech in the office that can handle the request. When he/she has done something with the request, worked on, or fixed. The techs are asked to input notes into their form so that it can be stored and later retrieved for quick reference of a problem.

The technician is required to close out each finished request by inputting the "fix" or solution of that particular request. Once the technician has successfully completed the task he/she will then submit the form once he/she have completed inputting all the pertinent information. A notification is then sent to the user via E-mail stating that the request has been completed. Figure 31 is the completed technician form once a request has been closed out.

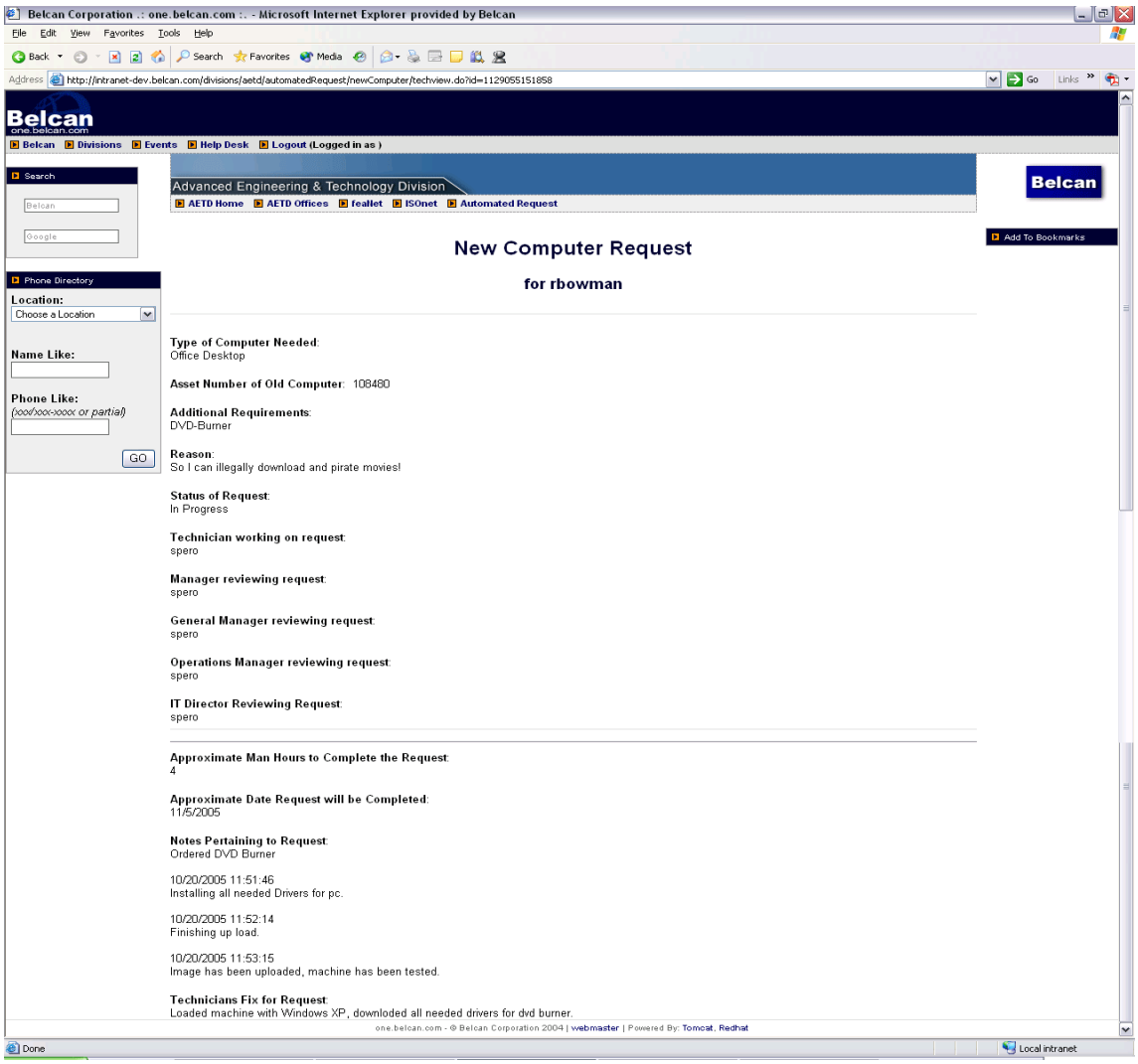


**Figure 31: Closed out Qualification Form**

The email below in Figure 32 is an example of the final email that is sent to the employee stating that the request has been completed by a specific technician. This is the end of the automated process.

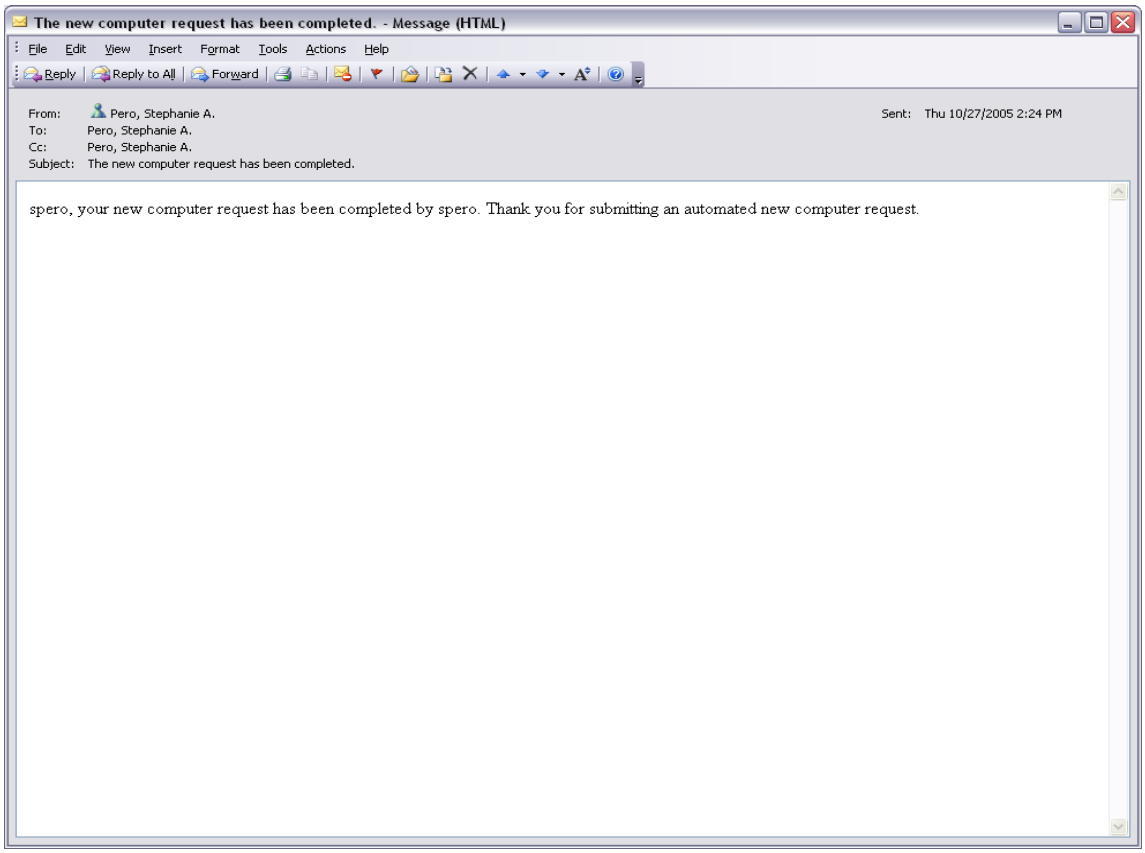


**Figure 32: Final Email to Belcan employee**

Finally, Figure 33 is the user checking their status one last time to ensure that the task was completed.

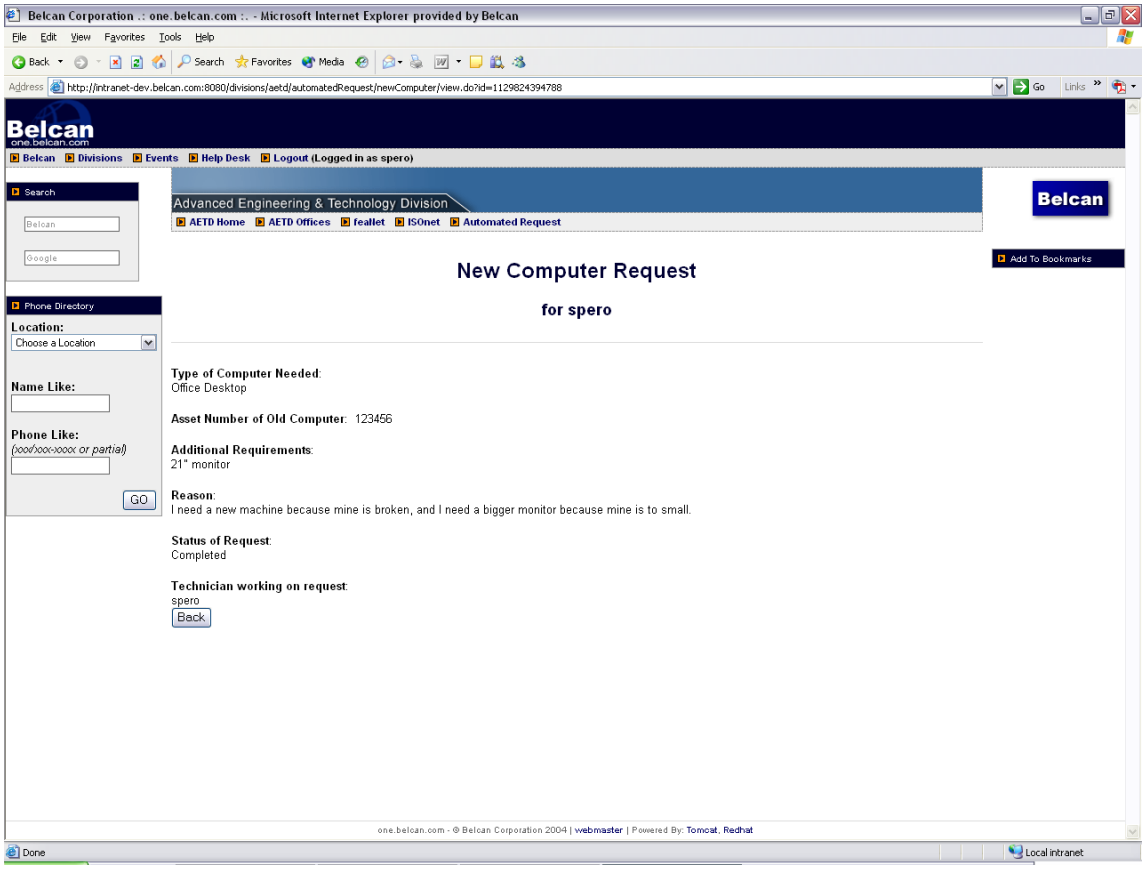


**Figure 33: Employee Checking Status**

### 6.2 Example of Other Functionality with Service Request Process

Each of the forms that are included in this web application follows the same process as the New Computer Request process shown in section 6.1. In this section I am going to demonstrate another functionality that was not present in the past version of this application for Senior Design II.

Each form allows for an employee to select from a drop down menu of pre-populated information, this particular form, the Service Request Form allows for employees to select a type of service that Belcan's technicians offer, but in any form user's will need something that is not on the list of items. So I have provided an "other" option to my drop down menu that allows for the employees to enter in an item that may not appear. This will enable Belcan employees a more versatile way of requesting services. Figure 34 is an example of what the Service Request form looks like.

As stated before an employee will get to this section of the application just as he or she would the New Computer Request Form.

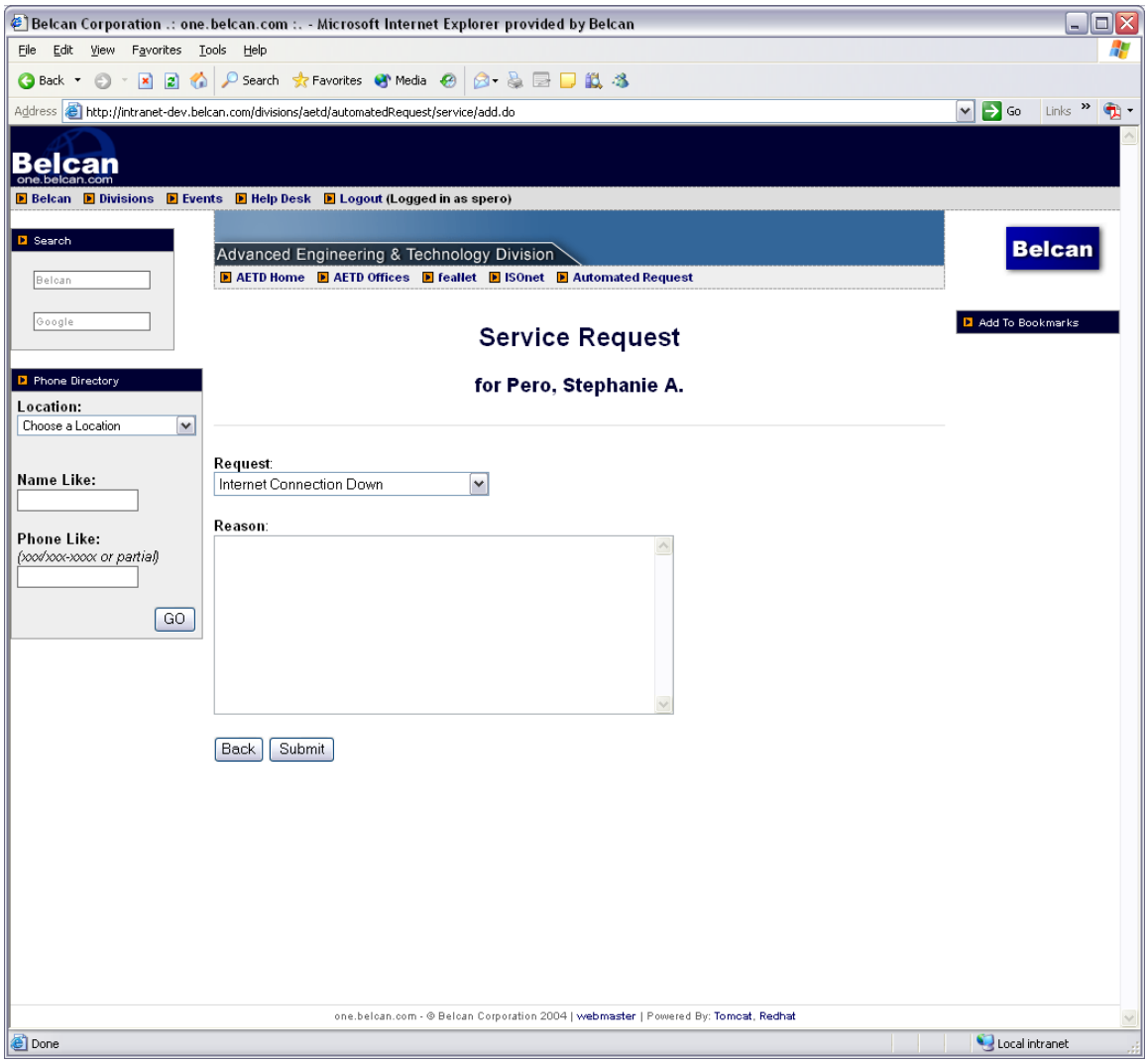


**Figure 34: Service Request utilizing "Other" Functionality**

Figure 35 below demonstrates how an employee can look through the original offered services and if the needed service is not present they choose the “other” option.

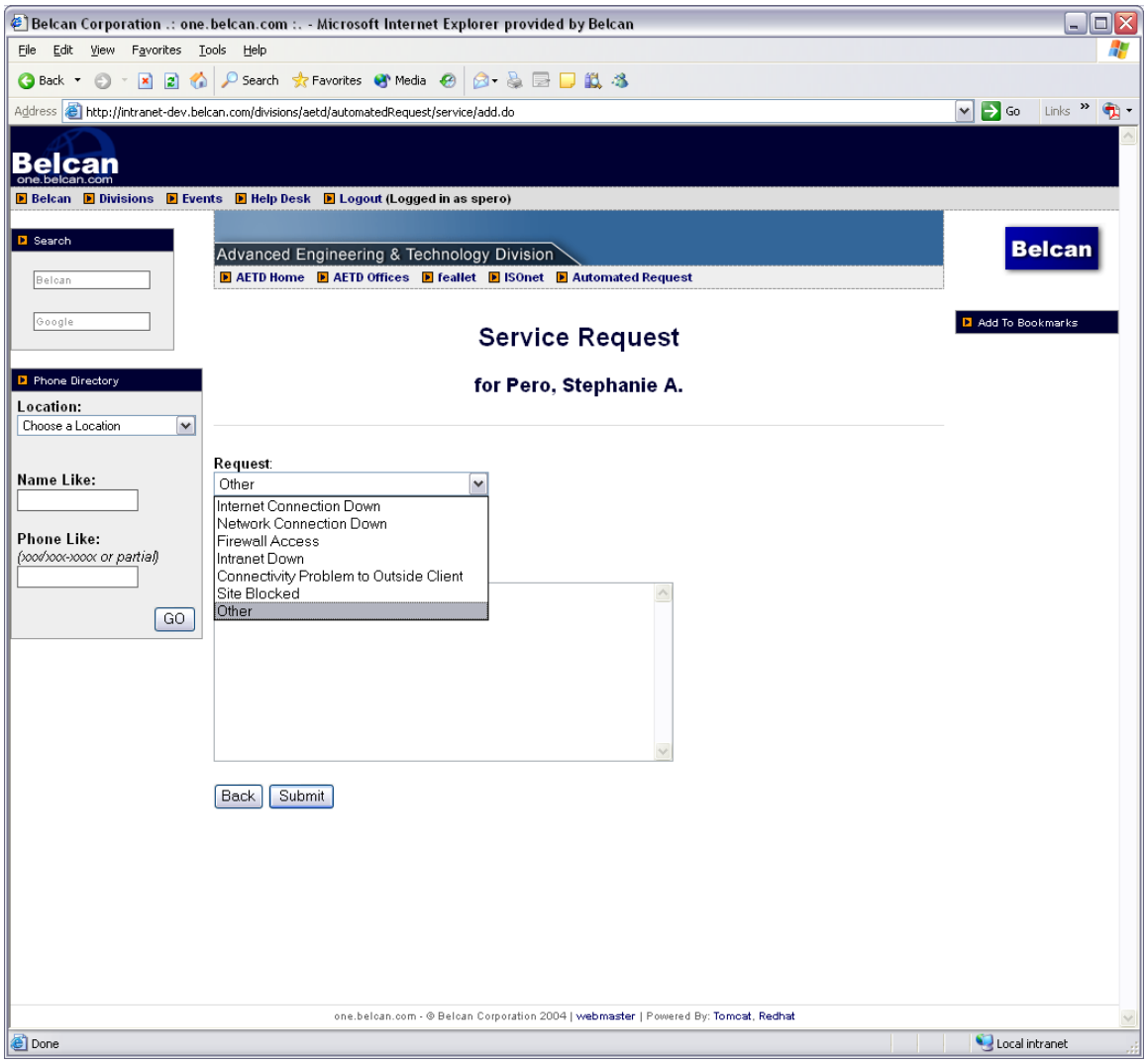


**Figure 35: Example of drop down menu with “other” option**

When the employee selects the “other” option a new text box appears that allows the user to input with he or she may need. This functionality is triggered by the employees selecting the other option, so if they were to select a different type of service the “other” input area would disappear. Figure 36 below represents this.

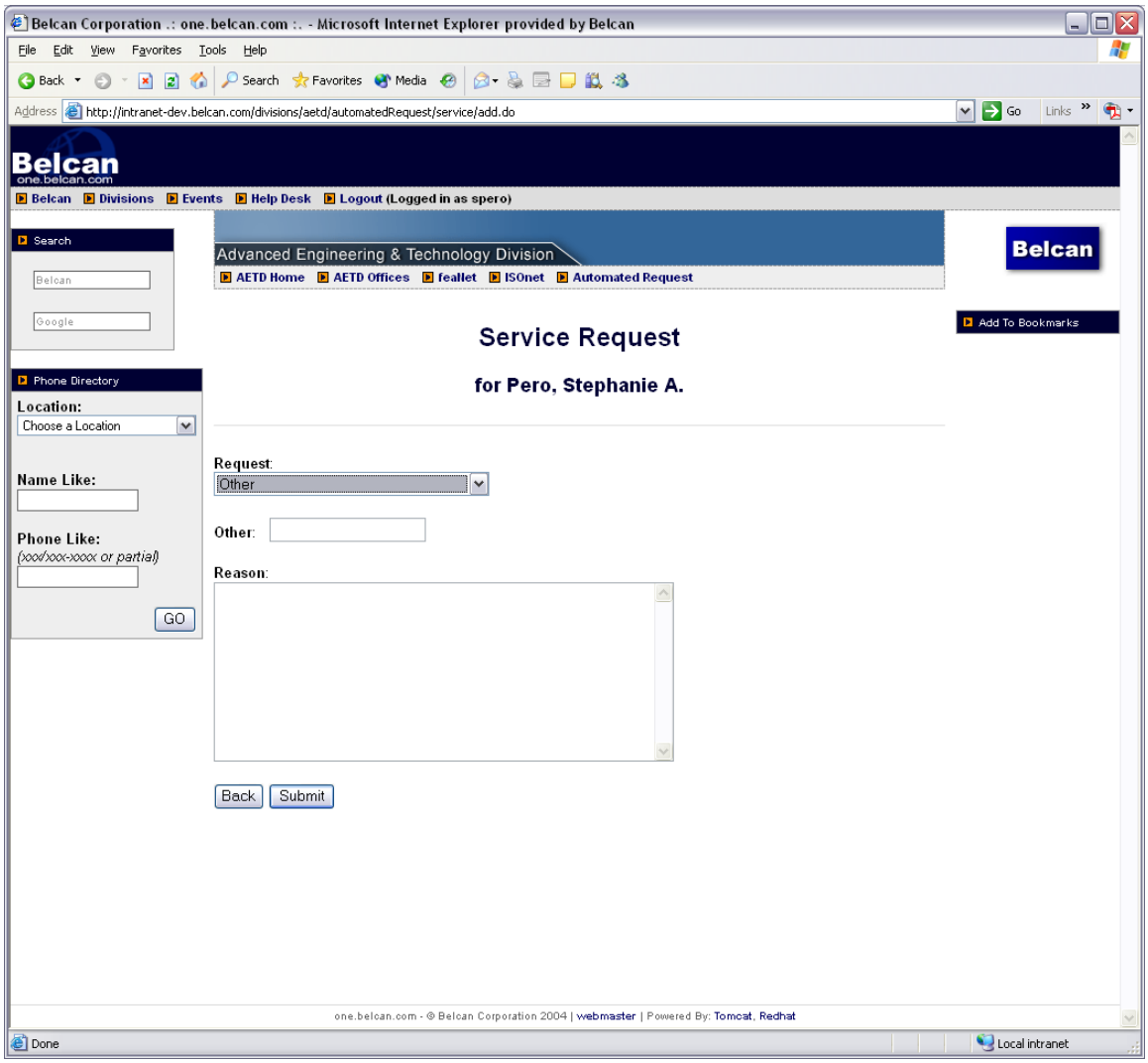


**Figure 36: Example of dynamic “other” option**

Once the “other” option is filled out the employee can move on and fill out the rest of the form. Figure 37 shows an example of a completed form.

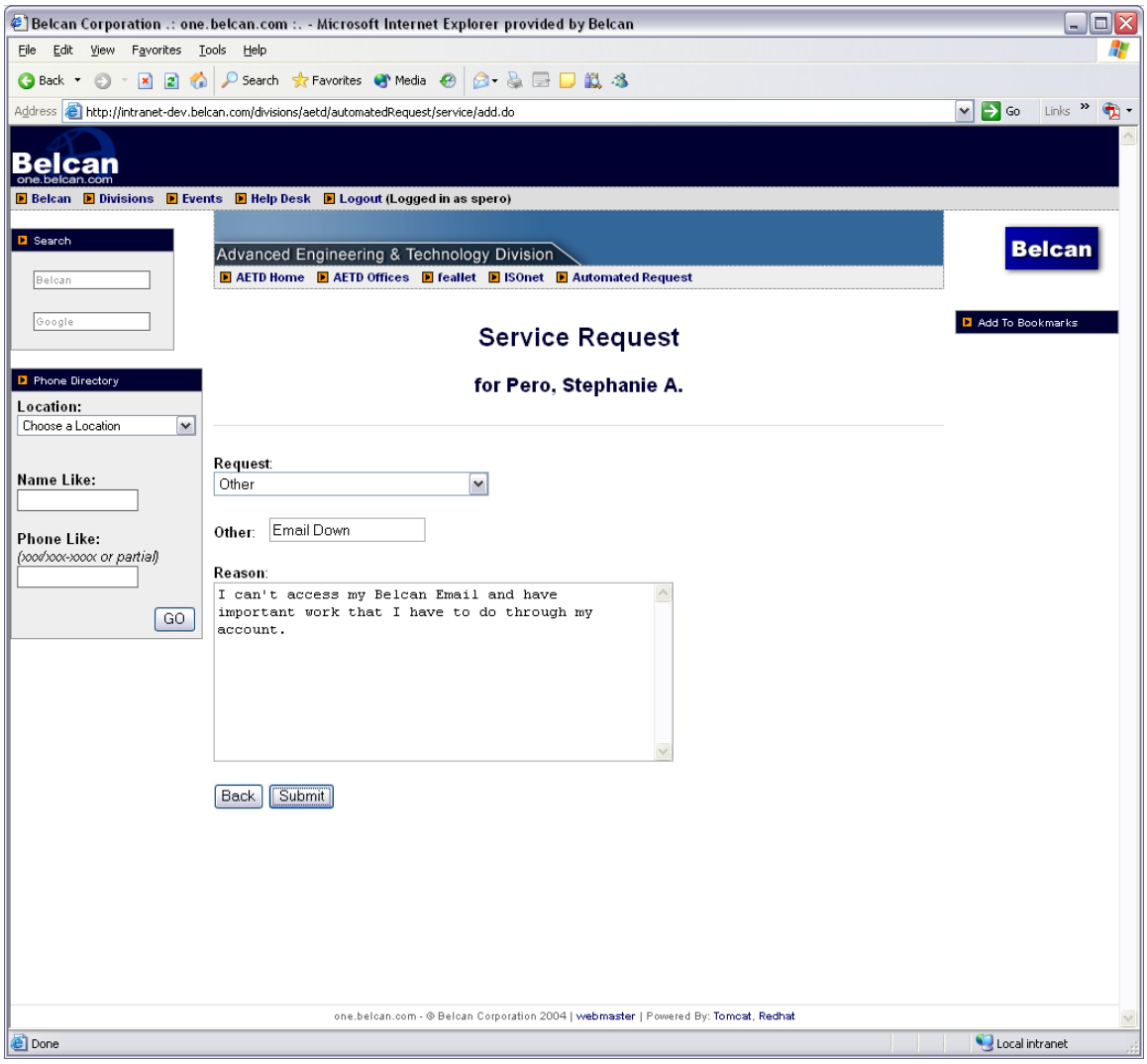


**Figure 37: Completed Service Form**

Once the user has submitted the form it is sent to a review page so that the employee can have one last look at the completed form before it is sent off for approval. Below in Figure 38 is an example of what the review page looks like. Notice that the input from the "other" option was retained in the review page. This information is sent to the database and stored for later retrieval.

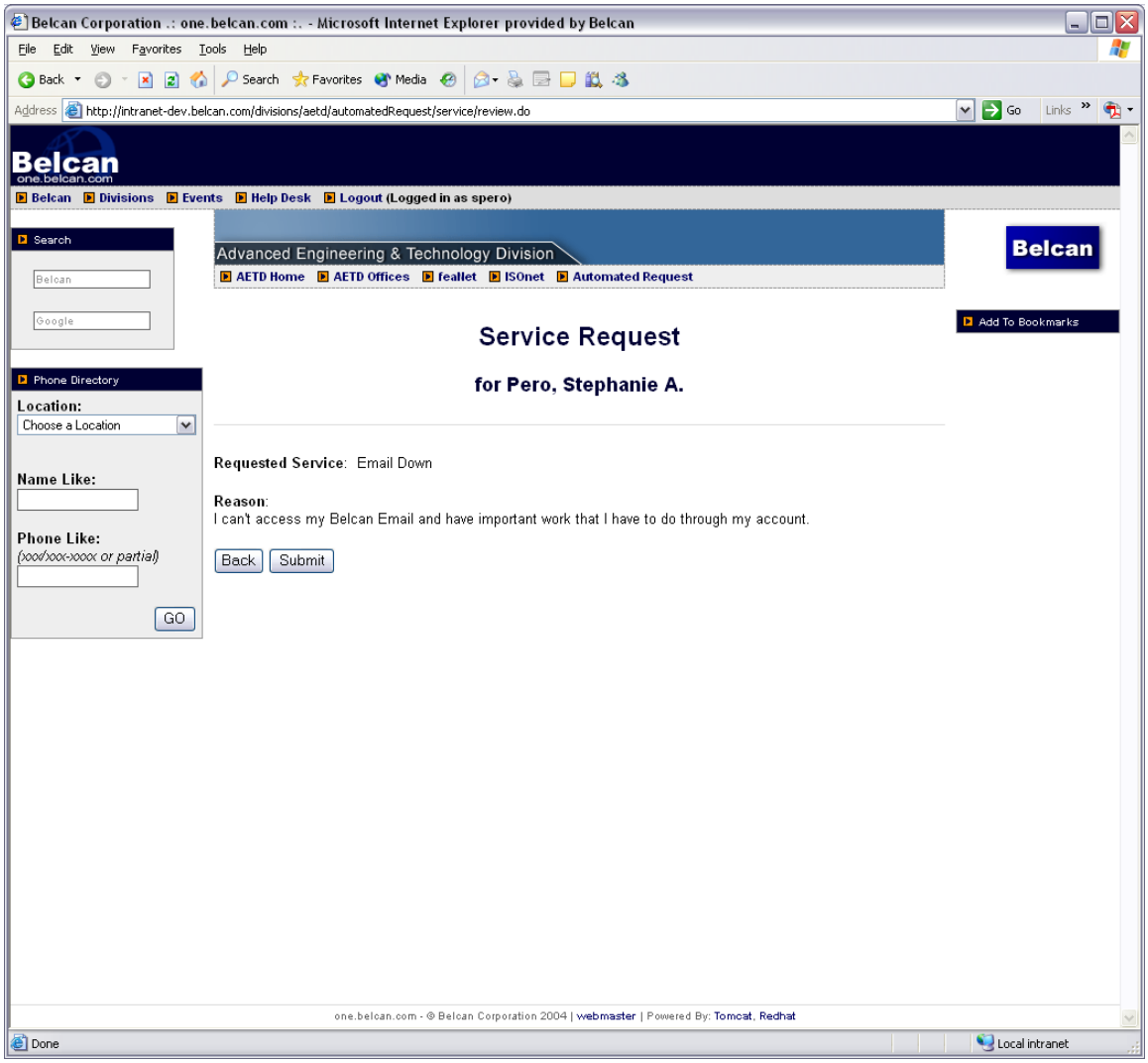


**Figure 38: Review Page showing retained "other" information**

### 6.3 Example of Denial Functionality with Software Request Process

During the life of a request each of the four types of managers must sign off or approve the request before it is sent off to the next person. If at any time during this process any of the managers decides that the request just is not feasible or is not a valid request they have the option to deny it and stop the process. This section demonstrates just that with the Software Request Form. This form is accessed through the same steps that the New Computer Request Form was accessed in section 6.1. The process for each of the forms is identical so all email notifications and forwards are the same for this and all forms in this application. Figure 39 is an example of what the Software Request Form looks like when completely filled out.

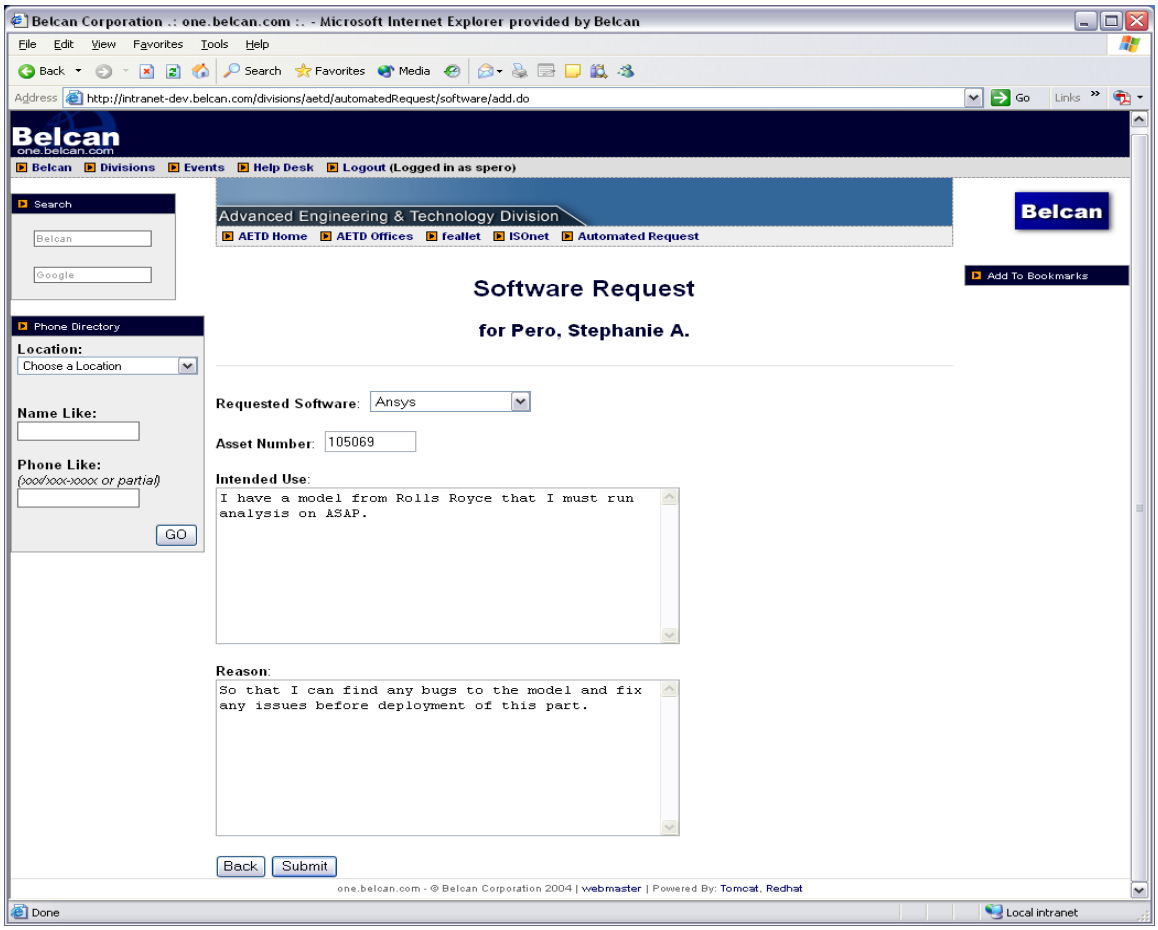


**Figure 39: Filled out Software request Form**

Once the Software Request form has been successfully filled out with no validation errors the employee is then redirected to the review page where he or she can review the information that they put into the form making sure everything is correct before it is sent off to the appropriate managers. Below in Figure 40 is an example of how the review page looks.

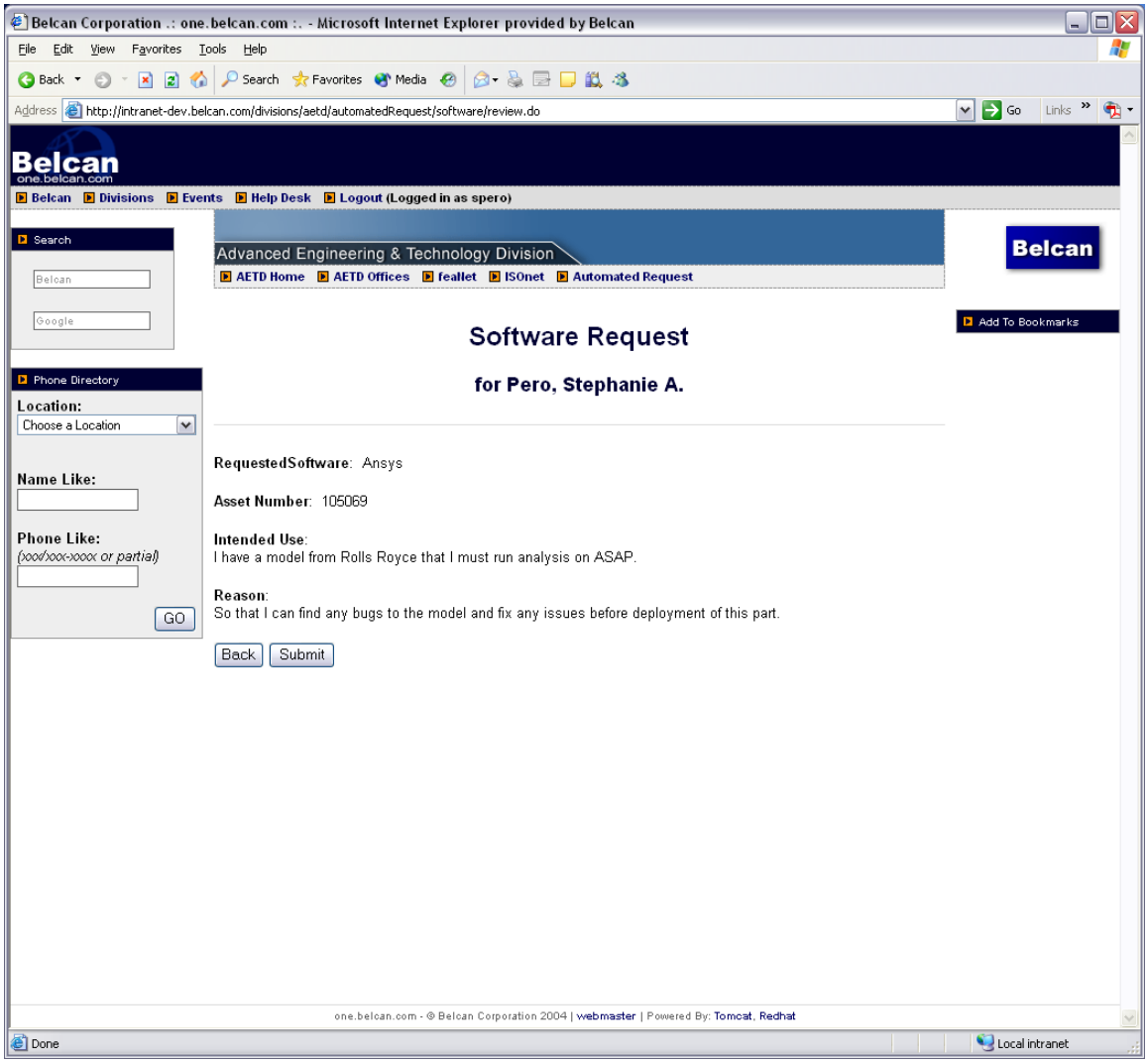


**Figure 40: Review Page of Software Request Form**

Once the employee has submitted the form to be approved by management an email is sent out to the Direct Manager of the employee providing a link to the completed request. When the link is followed the Direct Manager is directed to the approval/denial page, just as in all forms with this application. Below is an example showing that the Direct Manager has received the request and that they are approving it to be sent off to the General Manager.

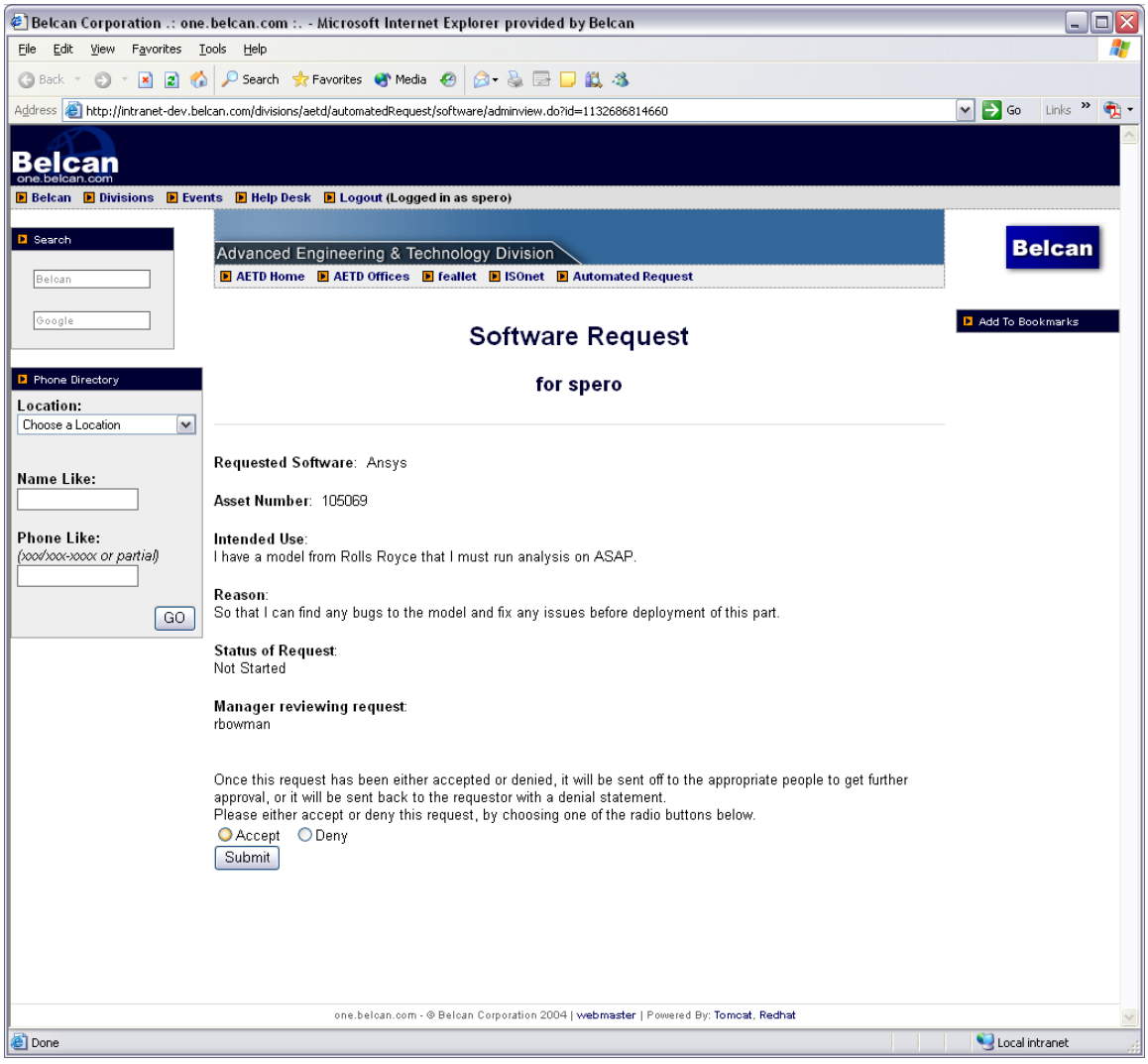


**Figure 41: Approval/Denial Page for Direct Manager**

Below in Figure 42 is an example of the General Manager's approval/denial page, but on this particular page the General Manager has decided to Deny the request.

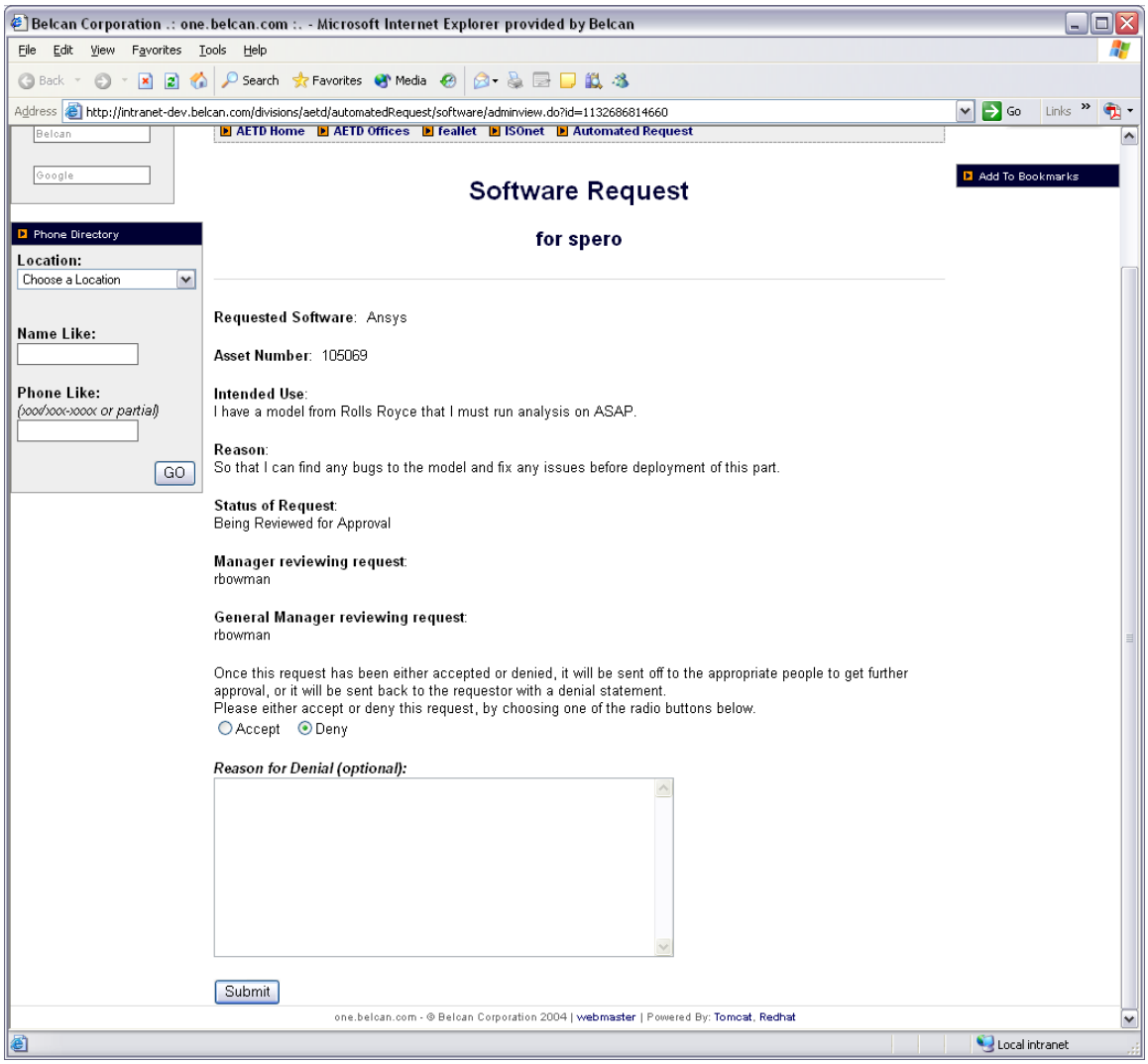


**Figure 42: Approval/Denial Page For General Manager**

When a request is denied a new textarea appears so that the denying manager can explain why the request is being denied if they feel the need to do so. Figure 43 is an example of a fully filled out Reason for Denial.

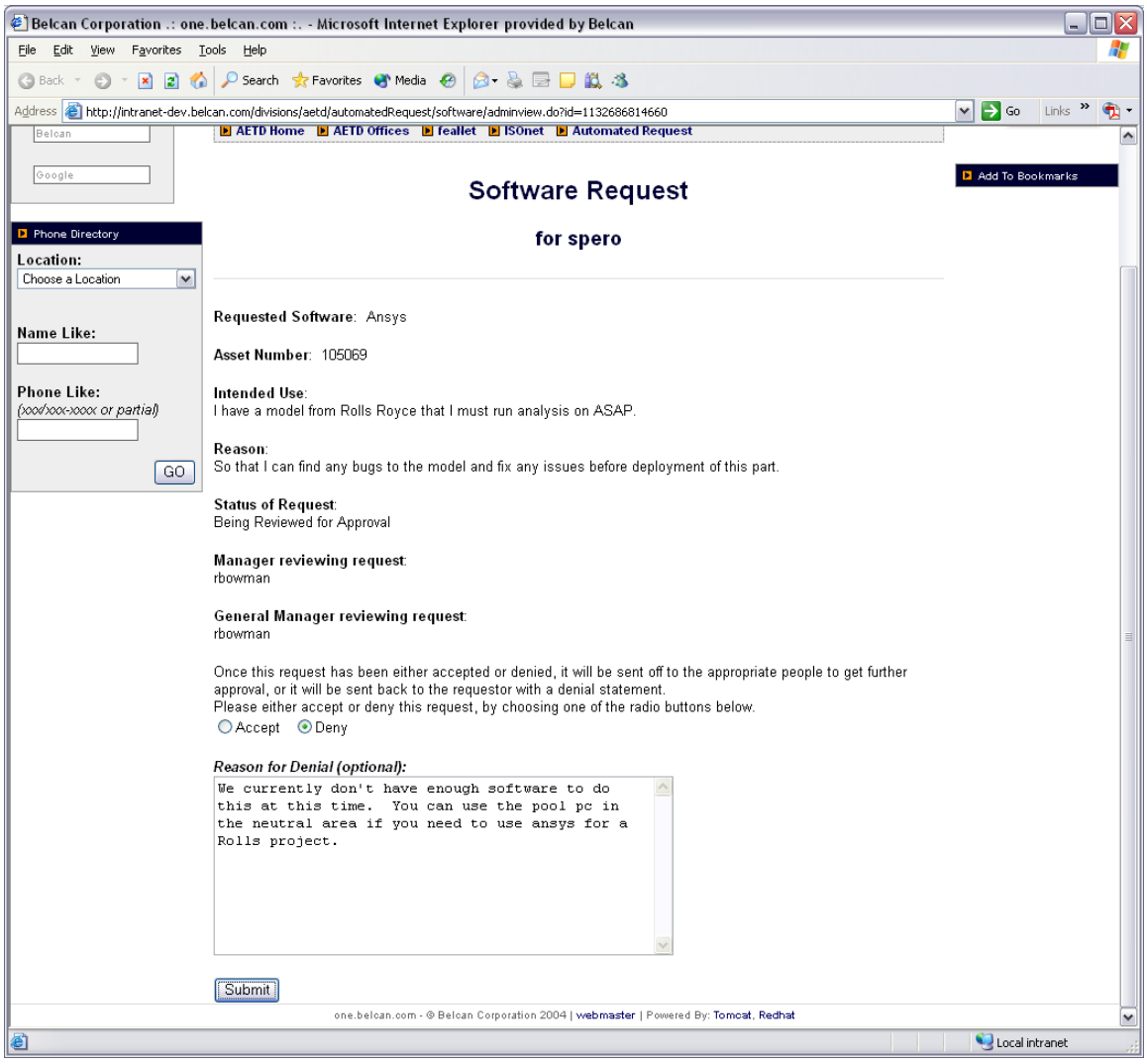


**Figure 43: Demonstrating the use of Reason for Denial when a request is denied**

Once the Manger submits this decision an email is sent to the user stating that the request was denied. This email offers a link to the status page of the request allowing the employee to view who denied and why they denied the request. This is demonstrated below in Figure 44.

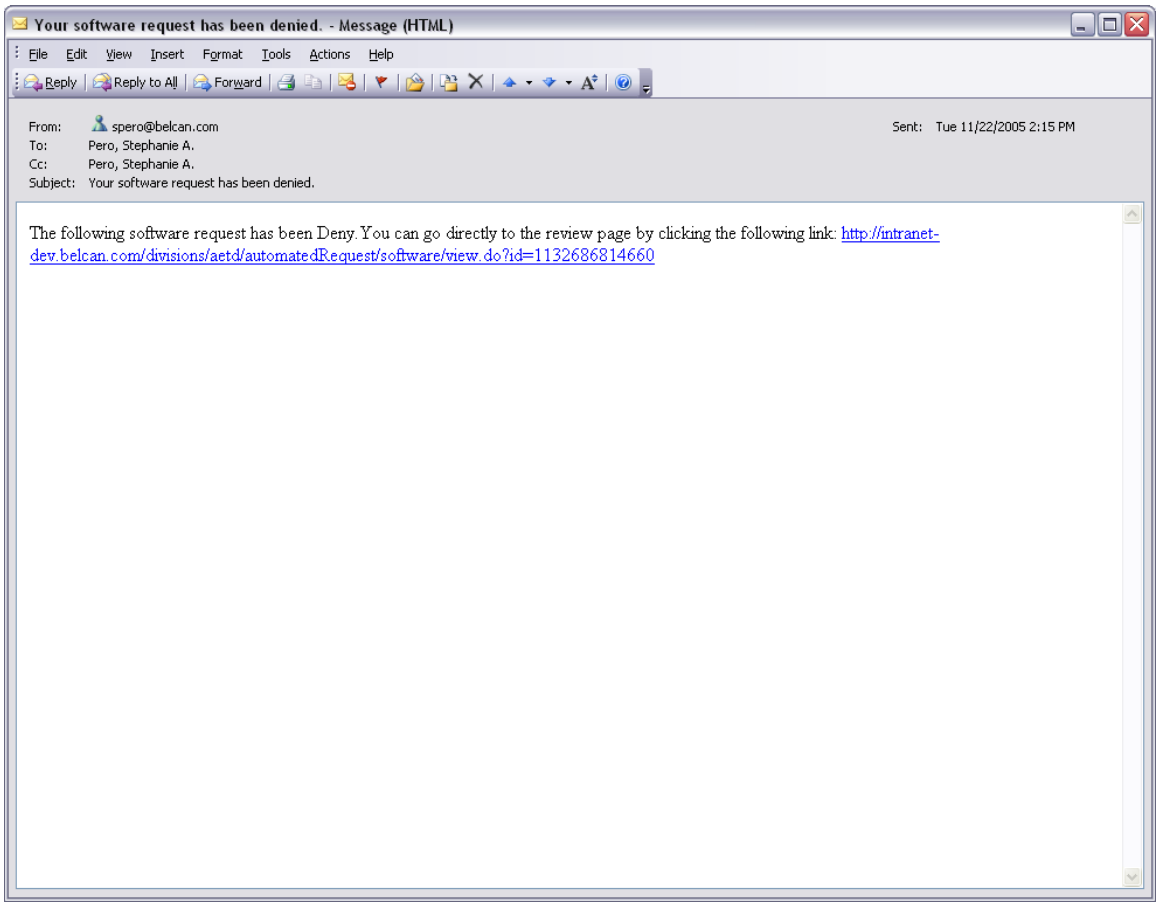


**Figure 44: Email to user stating that the request has been denied**

Below (Figure 45) shows what the employee sees when going to the status page when the request has been denied.

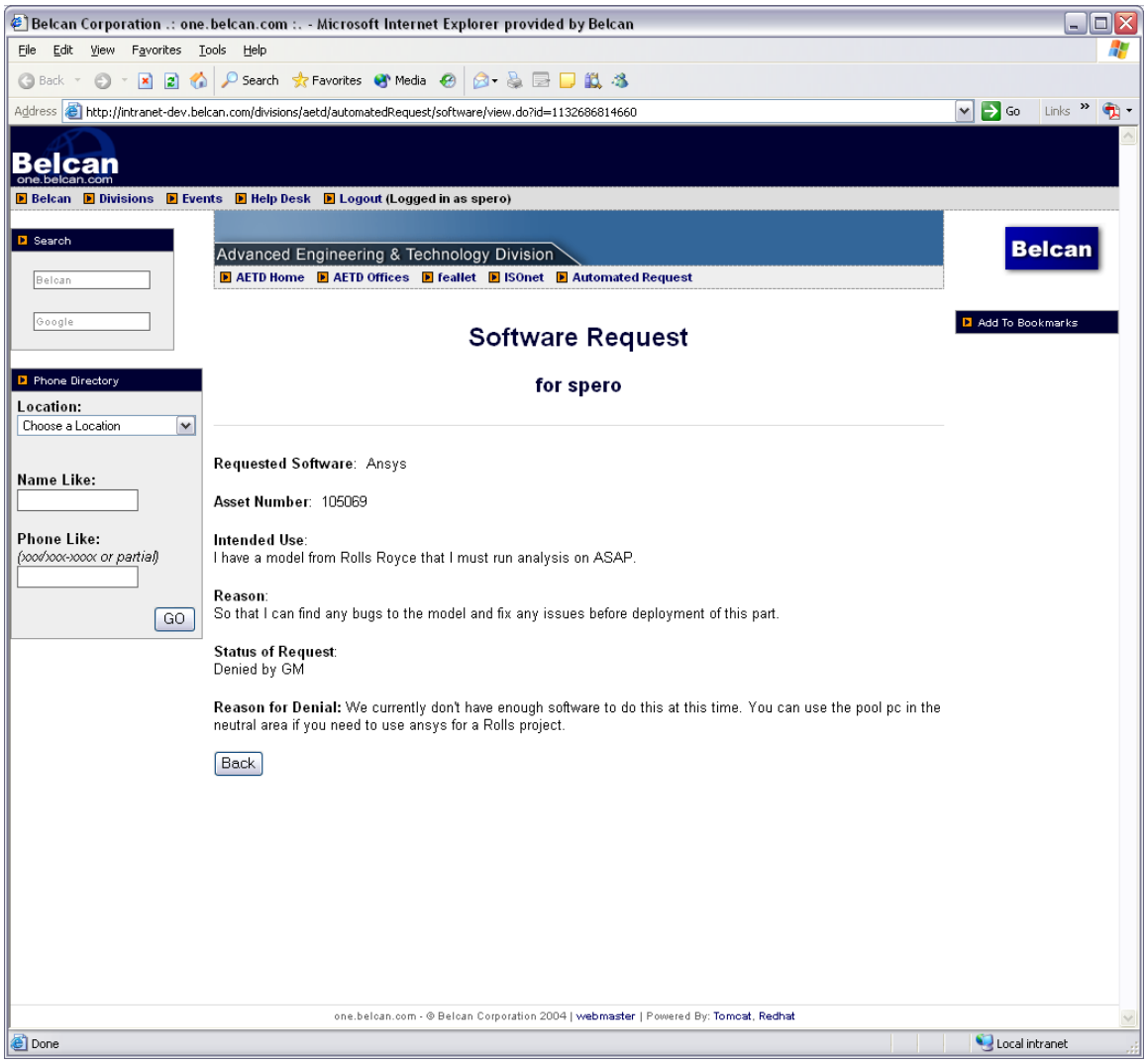


**Figure 45: User Status Page showing who denied the request and why**

# 7. Conclusions and Recommendations

## 7.1 Conclusions

With the addition of this Automated Request Process and Document Management system I feel that Belcan Corporation will benefit with a more up to date process. The old way of handling requests with hard paper copies works when the business is made up with a few employees in one office. Belcan on the other hand is a large business that has offices around the world and about 4,000 employees. The current processes also allows for multiple copies of the same form to be out on the Intranet, which in turn causes confusion on what form is up to date. Not only will this process better the communication between Belcan employees and Computer Services, but it will make the communications between the employees and their managers all the way up to the IT Director easier as well.

## 7.2 Recommendations

In creating any Web based application, it is vital to know the platform in which you are working with and to know the programming language very well. With using Java to do all of this particular application I found that it works great on any platform and it is very powerful. Knowing that a language has the potential to do so many different things is an important aspect in creating an application from virtually scratch.

Some of the other recommendations are to thoroughly investigate all third party software before stating the use of it in the application. I found that with Nutch, although it worked out for us, not all file extensions were searchable without additional plug-ins. Also, when I first found out about JUnit I initially thought this would be a plug-in that would go through and test my code. Again I was wrong in assuming this. JUnit is a

package that you must code in the testing classes for what you need it to do. This is a very customizable testing tool, but it is not what I had initially thought it was.

# Appendix A

**Use Case UC1: Request**
**Primary Actor:** Users

**Stakeholders and Interests:**
Users: Wants a fast and reliable solution to problem or request, with no errors.
Administrators: Want minimal errors or strain on the users while using the system that they created.
Company: Wants to accurately track all requests made by users, wants to make it fast and easy for users to make a request, and they want to have a system with as little problem as possible.

**Preconditions:**
Users must log into the system.

**Post conditions:** The user is identified, request is submitted, the system saves all information about request and user, the request is either accepted or denied, if accepted the request gets implemented and finally closed out. If denied the user is notified and everything stops.

**Main Success Scenario:**
1. User gets onto Intranet
2. Log into secured area
3. Go to correct section of site
4. User inputs information about request.
5. User then submits request for review.
6. If all is okay user then submits the request/if there are changes the user Edits the request then submits.
7. And Email is sent to the user's direct manager.
8. The Manger receives request.
9. The Manager accepts request, the request is then emailed to the General Manager.
10. The General Manager accepts request, the request is then emailed to the Operations Manager.
11. The Operations Manager accepts request, the request is then emailed to the IT Director.
12. The IT Director accepts request, the request is then emailed to the Tech.
13. Once the Tech receives the request, they qualify it by scheduling the approximated time and the approximated finish date.
14. The Tech then Implements the request.
15. Tech closes out the request once the implementation has been completed.
16. The user is then notified of the completion of the request.

**Alternative Flow:**

1. If the user cancels the request before it is sent to be qualified, no actions take place.
2. If the request makes it to the technician and it is not an appropriate request it is never qualified and no action takes place.
3. If the request makes it through the qualification but is not approved by the manager, general manager, operations manager, or the IT directory no further actions take place.
4. If the system fails at any point during the submissions of a new request, the user checks to make sure they are logged in, if that is true then they contact the help desk and the problem is then reported to the administrator of the system.

**Special Requirements:**

Everyone must have an account with Belcan to enter this part of the site.

**Frequency of Occurrence:**

This can happen as often as needed.  It is designed to be used as much as needed.

**Open Issues:**

-If the user goes directly to a request form will it automatically point them to the login page if they are not currently logged in, or will it throw an error?
-What should happen incase of a session time out?

**Use Case UC2: Accept/Deny Request**
**Primary Actor:** Manager, General Manager, Operations Manager, IT Director

**Stakeholders and Interests:**
Users: Have interest on whether or not their request is getting to the appropriate people and looked at in a timely manor.
Managers, General Managers, Operations Managers, and IT Director: Want a system that is fail proof and accurate as to whom it sends the information to.
Company: Wants to accurately track all requests made by users, wants to make it fast and easy for users to make a request, and they want to have a system with as little problem as possible.

**Preconditions:**
The Manager, General Manager, Operations Manager, and/or IT Directory must login to the system before accessing the form where they accept or deny the request.

**Post conditions:**
The Manager, General Manager, Operations Manager, and/or IT Director is identified, they accept or deny request and this "answer" is submitted, the system saves the response, the response is either sent to the next individual (First sent to the Manager then to the General Manager, then to the Operations Manager, and then off to the IT Director, back to the Operations Manager, then it is "assigned" to a Tech) or if denied the user is notified and everything stops, if accepted the request gets implemented by the Tech and finally closed out.

**Main Success Scenario:**
1. Manager, General Manager, Operations Manager, and/or IT Director gets onto Intranet
2. Follows the link that is provided in the email.
3. If not logged in, they log into the site
4. Manager, General Manager, Operations Manager, and/or IT Director then submits response for next reviewer.
5. If all is okay an email is sent to the user stating that the request has been accepted.
6. The Operations Manager then sends the request to the appropriate Tech to be implemented.
7. Once the Tech receives the request, they qualify it by scheduling the approximated time and the approximated finish date.
8. The Tech then Implements the request.
9. Tech closes out the request once the implementation has been completed.
10. The user is then notified of the completion of the request.

**Alternative Flow:**

1. If the user's Manager, General Manager, Operations Manager, and/or IT Director denies the request, everything stops.
2. If the request makes it to the technician and it is not an appropriate request it is never qualified and no action takes place.
3. If the system fails at any point during the submissions of an acceptance or denial, the Manager, General Manager, Operations Manager, and/or IT Director checks to make sure they are logged in, if that is true then they contact the help desk and the problem is then reported to the administrator of the system.

**Special Requirements:**

Everyone must have an account with Belcan to enter this part of the site.

**Frequency of Occurrence:**

This can happen as often as needed.

**Open Issues:**

-If the Manager, General Manager, Operations Manager, and/or IT Director goes directly to a request form from the email will it automatically point them to the login page if they are not currently logged in, or will it throw an error.
-What should happen incase of a session time out?

**Use Case UC3: Qualify Request**
**Primary Actor:** Tech

**Stakeholders and Interests:**
Users: Have interest on whether or not their request is getting to the appropriate people and looked at and implemented in a timely manor.
Managers, General Managers, Operations Managers, and IT Director: Want a system that is fail proof and accurate as to whom it sends the information to.
Tech: Wants a system that has accurate information so that they can process it and then implement it. Also they would like a fail proof system so that when they are inputting their notes and fixes the system will act as it is supposed to.
Company: Wants to accurately track all requests made by users, wants to make it fast and easy for users to make a request, and they want to have a system with as little problem as possible.

**Preconditions:**
The Technician must be logged into the system to view any request, input notes and fixes.

**Post conditions:**
The Technician is identified they then qualify the request by inputting the approximated hours, and the approximate finish date. They then take this time to implement the request, once this has been completed the tech closes out the request and the user is notified.

**Main Success Scenario:**
1. Technician gets onto Intranet
2. Follows the link that is provided in the email.
3. If not logged in, they log into the site
4. Technician then qualifies request by scheduling the approximated time and the approximated finish date.
5. Tech implements request
6. Tech then submit notes and/fix.
7. Tech closes out the request once the implementation has been completed.
8. The user is then notified of the completion of the request.

**Alternative Flow:**
1. If the request makes it to the technician and it is not an appropriate request it is never qualified and no action takes place.
2. If the system fails at any point during the submissions of a new request, the user checks to make sure they are logged in, if that is true then they contact the help desk and the problem is then reported to the administrator of the system.

**Special Requirements:**
Everyone must have an account with Belcan to enter this part of the site.

**Frequency of Occurrence:**
This can happen as often as needed.

**Open Issues:**
-If the Tech goes directly to a request form will it automatically point them to the login page if they are not currently logged in, or will it throw an error?
-What should happen incase of a session time out?

**Use Case UC5: Closeout Request**
**Primary Actor:** Tech

**Stakeholders and Interests:**
User: Want to know as soon as possible when the request has been implemented and completed.
Managers, General Managers, Operations Managers, IT Director: Want a system that is fail proof and accurate as to whom it sends the information to and what information that they are receiving.
Tech: Wants a system that has accurate information so that they can process it and then implement it. Also they would like a fail proof system so that when they are inputting their notes and fixes the system will act as it is supposed to. Also, techs want to know when they do complete a request that the correct information is getting sent to the correct user and managers.
Company: Wants to accurately track all requests made by users, wants to make it fast and easy for users to make a request, and they want to have a system with as little problem as possible.

**Preconditions:**
The Tech must be logged into the system in order to close out a request,

**Post conditions:**
The Technician is identified, they then input what the fix was to the request, and hit the submit button. This is all saved and the user is notified of the completion.

**Main Success Scenario:**
1. Technician gets onto Intranet
2. Log into the site
3. Go to their main view page
4. They access the appropriate request
5. Makes sure it has been implemented
6. Tech then submits the fix.
7. Tech closes out the request.
8. The user is then notified of the completion of the request.

**Alternative Flow:**
1. If the system fails at any point during the submissions of a new request, the user checks to make sure they are logged in, if that is true then they contact the help desk and the problem is then reported to the administrator of the system.

**Special Requirements:**
Everyone must have an account with Belcan to enter this part of the site.

**Frequency of Occurrence:**
This can happen as often as needed.

**Use Case UC6: Manage Site**
**Primary Actor:** Administrator

**Stakeholders and Interests:**
User: Want to know as soon as possible when the request has been implemented and completed.
Managers, General Managers, Operations Managers, IT Director: Want a system that is fail proof and accurate as to whom it sends the information to and what information that they are receiving.
Tech: Wants a system that has accurate information so that they can process it and then implement it. Also they would like a fail proof system so that when they are inputting their notes and fixes the system will act as it is supposed to. Also, techs want to know when they do complete a request that the correct information is getting sent to the correct user and managers.
Company: Wants to accurately track all requests made by users, wants to make it fast and easy for users to make a request, and they want to have a system with as little problem as possible.
Administrator: Has an interest that the site has been built well and that they are not getting a lot of calls about errors.

**Main Success Scenario:**
There are no errors.

**Alternative Flow:**
1. There are errors and the Administrator must go and figure out why the system has failed.
2. The Administrator then fixes the problem.
3. Tests the solutions
4. Updates the Live site.

**Frequency of Occurrence:**
Hopefully no problems occur but if they do it is minimal.

**Open Issues:**
-If anyone goes directly to a request form will it automatically point them to the login page if they are not currently logged in, or will it throw an error?
-What should happen incase of a session time out?

## Appendix B
## New Computer Request Form Examples

### B1 Example of Validation for New Computer Request

In order for Belcan user's to use an automated request process, they must have access to a web form to start the process. I implemented the use of Struts so that I could have separation between presentation and content. So with the use of some simple Struts tools to incorporate some cool technology with some basic html formatting code. I use the tag <html:errors/> to reference a properties file that is used by the entire Intranet. An example of the error entries that I added into the file include:

```
errors.header=<font color="red"><h3>Validation
Error</h3>The following errors were found.  Please
correct them.<br><ul>
errors.footer=</ul></font>
error.oldCostCenter.required=<li>The old cost center
is required.</li>
error.newCostCenter.required=<li>The new cost center
is required.</li>
error.jackNumberFrom.required=<li>The jack number from
is required.</li>
error.primaryUserFrom.required=<li>The primary user
from is required.</li>
error.jackNumberTo.required=<li>The jack number to is
required.</li>
error.primaryUserTo.required=<li>The primary user to
is required.</li>
error.date.required=<li>The date is required.</li>
error.assets.required=<li>The asset number of your
machine is required, and it must be 6 numbers long
with no characters.</li>
error.reason.required=<li>The reason is required.</li>
```

These lines of code are used in unison with code from the new computer request class file that set constraints on the user input for the necessary fields. This section of code ensures that the information that the user inputs into the form fields is valid depending on the data types. In the first test I am checking to see if the field "type" is

null or if it is less than 1 or equal to nothing. If this test returns true then an error is thrown and the message from the above code is shown. The last test that is done here is one that is a little different than the rest in such a way that I am testing for several things. First, I am testing to see if the input is null or the length is not equal to 6. If that test fails it throws the normal validation errors, but if it passes that it must go through another check to see if the input is an integer. If that check fails then the error message is then displayed, if it passes then the form can be submitted to the review page.

```
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request)
{
	ActionErrors errors = new ActionErrors();
	if( type == null || type.length() < 1)
	{
		errors.add("type", new ActionError("error.type.required"));
	}
	if(reason == null || reason.length() < 1)
	{
		errors.add("reason", new ActionError("error.reason.required"));
	}
	if(asset == null || asset.length() != 6)
	{
		errors.add("asset", new ActionError("error.asset.required"));
	}else{
		try
		{
			Integer.parseInt(asset);
		}
		catch(NumberFormatException e)
		{
			errors.add("asset", new ActionError("error.asset.required"));
		}
	}
	return errors;
}
```

## B2 Email example

In this application there are certain functionalities that make this system work proficiently and dynamically. One of these that may be most important for this application happens to be the email notification. This simple function is what makes this application such a robust and dynamic system. In the example below I have the entire function that causes the notification. First, I must create a String with the name emailbody so that I can have a message within the email being sent. The first six lines of code are all the same as far as what they do. To create a new line within the email I have called the string again and used the += symbols to append the next line of text to the email body. Once the body has been completed I move on to the creation of the subject line. This too is created as a string. Next I create a new session which gather's the submitting user's information. Then I set the; to, from, and copy variables so that when the send function is finally reached all the parameters have been filled in. I use the user's information from the session to fill in the email address of the user. Finally I call the one line of code that does all the work.

```
//email appropriate people
String emailbody = "The following new computer request has been submitted for
approval.  ";
emailbody += "You can go directly to the review page by clicking the following link:";
emailbody += " <a href=\"http://" + req.getServerName() +
":8080/divisions/aetd/automatedRequest/newComputer/adminview.do";
emailbody += "?id=" + id + "\">";
emailbody += "http://" + req.getServerName() +
":8080/divisions/aetd/automatedRequest/newComputer/adminview.do?id=" + id;
emailbody += "</a>";
String emailsubject = "A new, new computer request has been submitted.";

        HttpSession session = req.getSession();

        User user = (User) session.getAttribute("user");
        String[] to = {"rbowman@belcan.com"};
```

```
                String from = user.getEmail();
                String[] cc = {"spero@belcan.com"};

//This email will be first sent to the direct manager
Email.sendToWithCC(to, from, cc, emailsubject, emailbody);
```

With all this put together I am able to send emails automatically to employee's direct managers all the way up to the IT Director without the submitting employee knowing or needing to know who it is getting sent to.

# Appendix C

## Peripheral Request Processes

Once an employee tries to access one of the forms or their home page, they are then redirected to a login screen so the authentication process can start and so that active directory can retrieve user information. Below are a few examples of a user logging into the system (Figures 46 and 47).

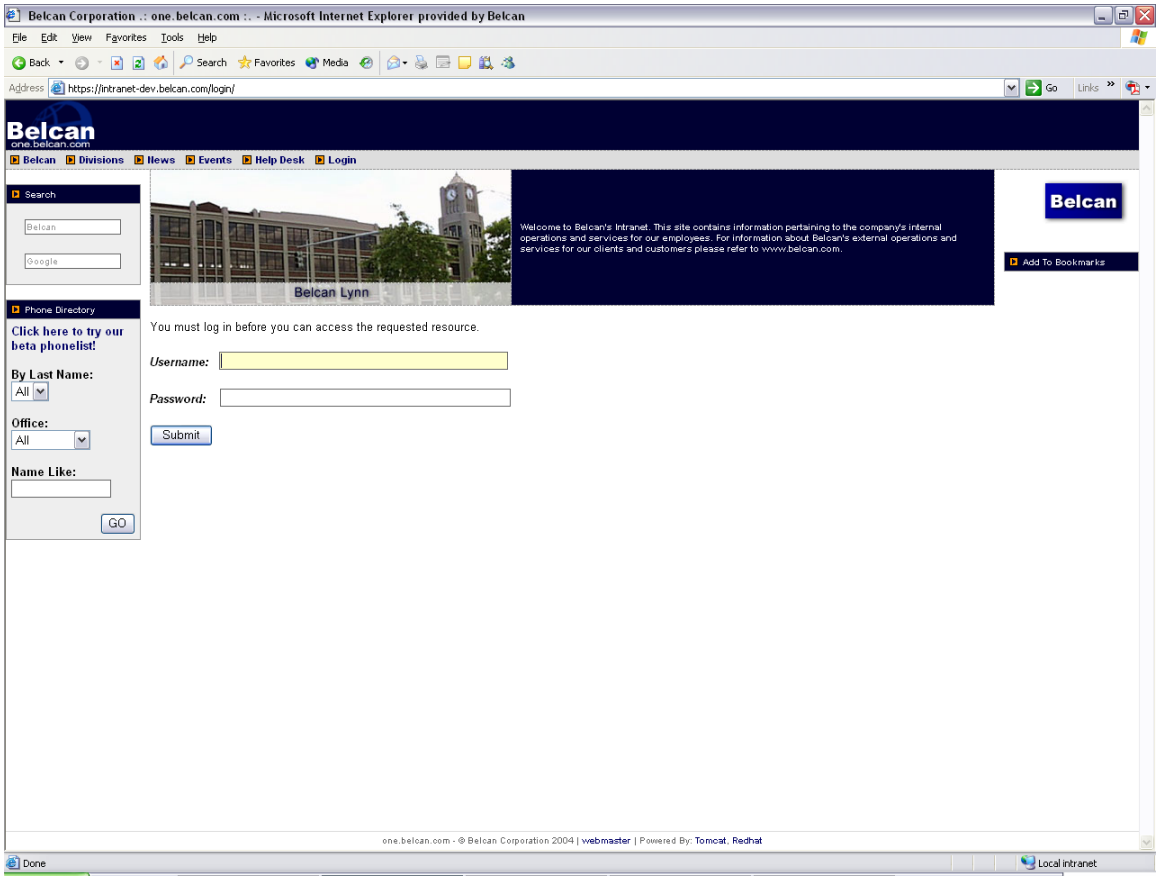


**Figure 46: Belcan Corporation's Login Screen**

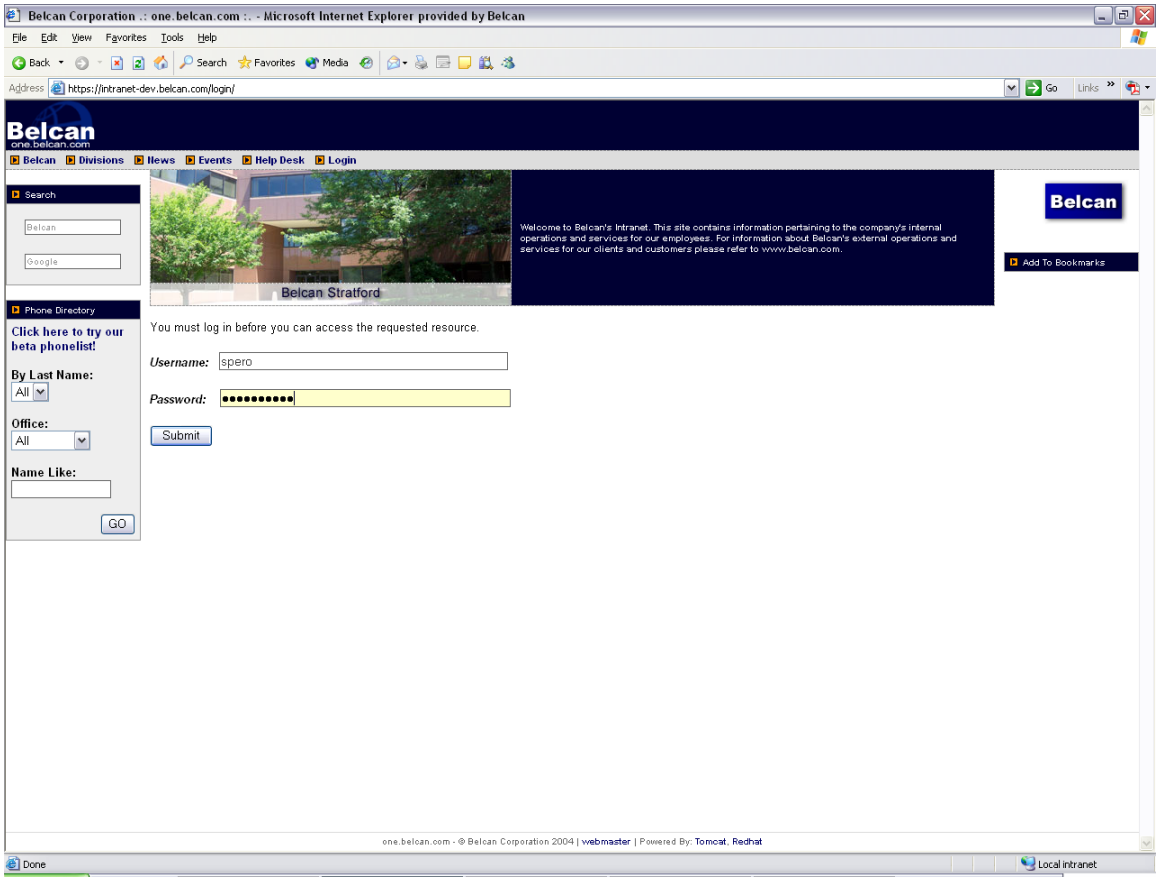


**Figure 47: User Logging in**

When the employee tries to access a part of the site that is restricted they are automatically brought back to the login screen. The code that is responsible for this action is as follows:

```
if(!CheckUserLogin.isUserLoggedIn(req))
{
        return mapping.findForward("login");
}
```

This block of code simply checks if the user has logged in or not. If they are not logged in then the check returns true and forwards the user to the login page. This CheckUserLogin.isUserLoggedIn references a Struts action that accesses a java class that checks LDAP for user authentication. If the user is logged in then the return value will

be a success which in turn forwards the user to the desired place on the site. This is the java class that does this work:

```
public class CheckUserLogin extends Action
{
public ActionForward execute(ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response)
{
          if(isUserLoggedIn(request))
          {
           return mapping.findForward("success");
          } else {
           return mapping.findForward("login");
          }
 }

 public static boolean isUserLoggedIn(HttpServletRequest request)
 {
  if(LdapLogin.isUserLoggedIn(request))
  {
   return true;
  }else{
   HttpSession session = request.getSession();
   StringBuffer url = new StringBuffer();
   url.append(request.getRequestURL());
   Enumeration attrNames = request.getParameterNames();

   if(attrNames.hasMoreElements())
   {
      url.append("?");
      while(attrNames.hasMoreElements())
      {
         String attrName = (String)attrNames.nextElement();
         url.append(attrName);
         url.append("=");
         url.append(request.getParameter(attrName));
         if(attrNames.hasMoreElements())
         {
            url.append("&");
         }
      }
   }
   session.setAttribute("uri", url.toString());
   return false;
  }  }}
```

Once the user is logged in they are then redirected to their original destination. In this case they were redirected to the Peripheral Request Form, shown below in Figure 48.

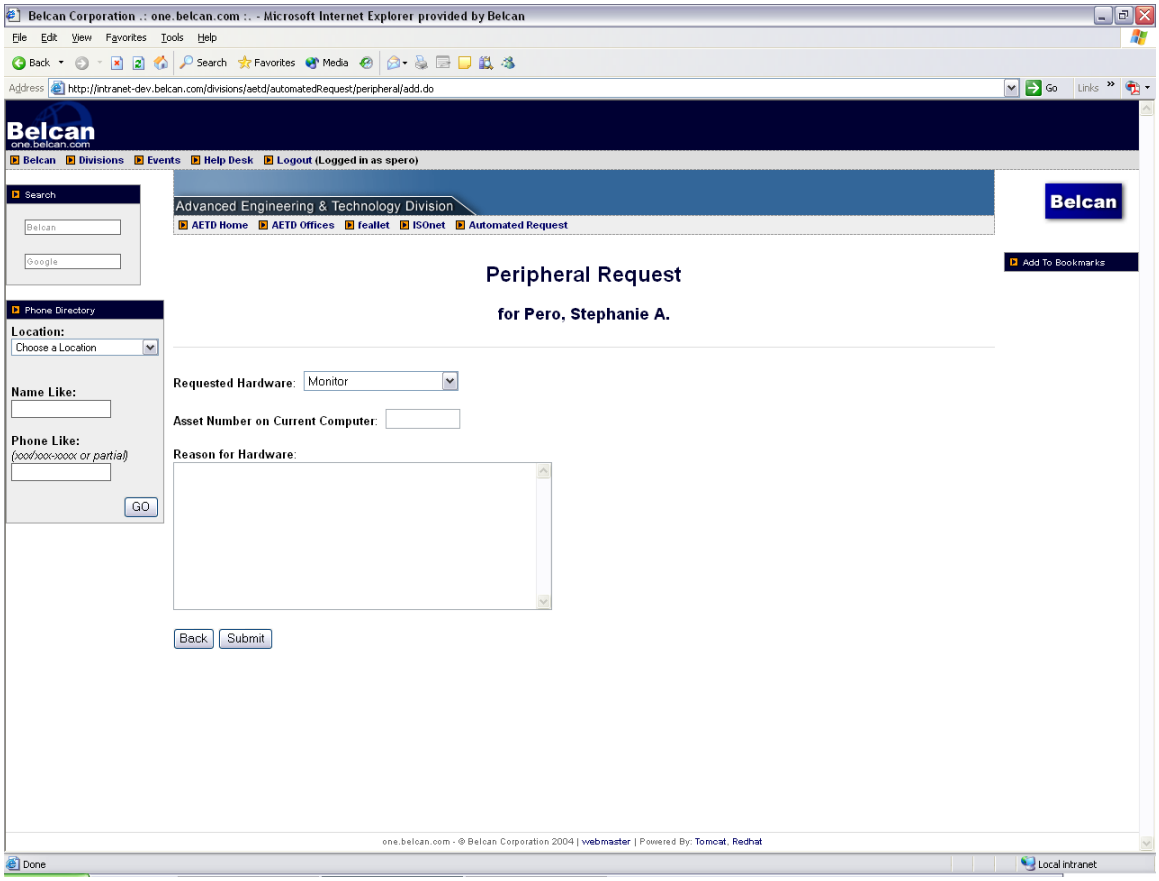


**Figure 48: Peripheral Request Form**

The code below is the actual jsp behind this form. This is the presentation layer so this code does not care how the content layer retrieves the information, it is just responsible for displaying it. This code also utilizes struts tags and a struts action. The struts tags are all of the <html:…> tags. You will notice that with normal html the text area's and textboxes have names, here in struts, property is used in names place. This offers the same functionality but allows communication between struts content and presentation.

The struts action that is preformed from this form is located in the form tag; <html:form action="/divisions/aetd/automatedRequest/peripheral/review">. This is how Struts connects the two layers. This is actually retrieved and used in another file in which I will show later in this section, but for now remember this is the link between the presentation and content layers.

Here I include the admin header which allows for user login.

```
<%@ include file="/includes/adminAetdHeader.jsp" %>
<%@ include file="hardware.jsp" %>

<jsp:useBean id="hardware" type="java.util.List" scope="page"/>
<script language="javascript">
<!--
 function displayOther()
 {
        if(document.PeripheralForm.requestedHardware.value == "Other")
        {
                var otherbox = document.getElementById("td1");
                otherbox.className = "visible";
                document.PeripheralForm.other.disabled = false;

        }
        else
        {
                var otherbox = document.getElementById("td1");
                otherbox.className = "hidden";
                document.PeripheralForm.other.disabled = true;
                document.PeripheralForm.other.value = "";
        }
 }
-->
</script>
```

Here I am displaying the Form name along with the current logged in user's name. The users name is derived from a session created when the user logs in.

```
<center>
        <h1>Peripheral Request</h1>
        <h2>for <c:out value="${user.name}"/></h2>
</center>
```

<html:errors> is the tag that will create the validation messages when a validation error occurs in the form.

```
<html:errors/>
```

This next set of tags are struts specific and are just like those of the normal html tags except that with struts we have to explicitly state that we are using html formatting. So here we are setting the action of the submit button to redirect along with all the valid input information to the review page.

```
<html:form action="/divisions/aetd/automatedRequest/peripheral/review">
<hr noshade color="#DDDDDD" size=1>
```

This next block of code is how I set the value of the form fields.

```
<p>
        <strong>Requested Hardware</strong>: 
        <html:select property="requestedHardware" onchange="displayOther();"
styleClass="visible">
        <html:options collection="hardware" property="desc" labelProperty="desc"/>
        </html:select>
        <p class="hidden" id="td1">
        <strong> Other</strong>:  
        <html:text property="other" styleId="othersID" size="20" disabled="true"/>
<p>
<p>
        <strong>Asset Number on Current Computer</strong>: 
        <html:text property="asset" size="10"/>
  <p>
        <strong>Reason for Hardware</strong>:<br>
        <html:textarea property="reason" rows="10" cols="50"/>
<p>
```

This button redirects the user back to the index page.

```
<html:button property="" value="Back"
onclick="window.location='/divisions/aetd/automatedRequest/index.do'"/>
```

This button is what actually invokes the action from the opening form tag and redirects the page to the review page.

```
<html:submit property="" value="Submit"/>
```

This hidden tag is again the same type of tag that is used in html, it allows for the value of the user's username to be forwarded along with the other information to the next page.  This is what I do all of my retrievals with.

```
<input type="hidden" name="username" value="<c:out value="${user.username}"/>">
```

Here I am closing out the form tag.

```
</html:form>
<script language="javascript">
 if(document.PeripheralForm.other.value != "" && document.PeripheralForm.other.value
!= null)
  {
   var td1 = document.getElementById("td1");
   td1.className = "visible";
   document.PeripheralForm.other.disabled = false;
  }
</script>
<%@ include file="/includes/footer.jsp" %>
```

Below is an example of an employee looking through the different options that are dynamically pulled from the database.

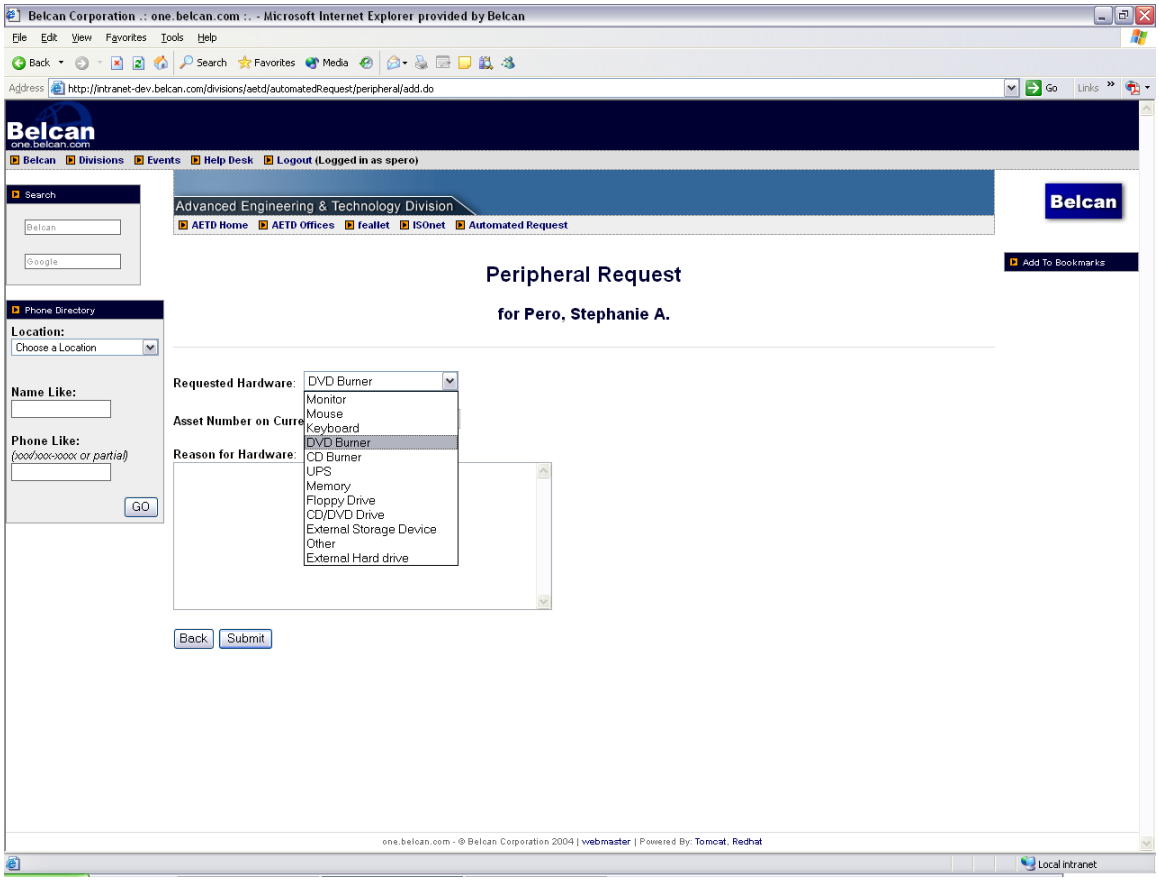


**Figure 49: Peripheral Request drop down menu of available hardware**

In the figure above I am pulling dynamic information from a database to populate the information in the drop down menu. Below is the code that is located in the jsp that is responsible for this action.

```
<html:select property="requestedHardware" onchange="displayOther();"
styleClass="visible">
        <html:options collection="hardware" property="desc"
        labelProperty="desc"/>
</html:select>
```

This snippet simply uses the bean and include from above;

```
<%@ include file="hardware.jsp" %>
<jsp:useBean id="hardware" type="java.util.List" scope="page"/>
```

and retrieves the information from the list.

When the employee is finished filling out the form they will submit the form and then get redirected to a review page. Below is an example of the Peripheral Form filled out.

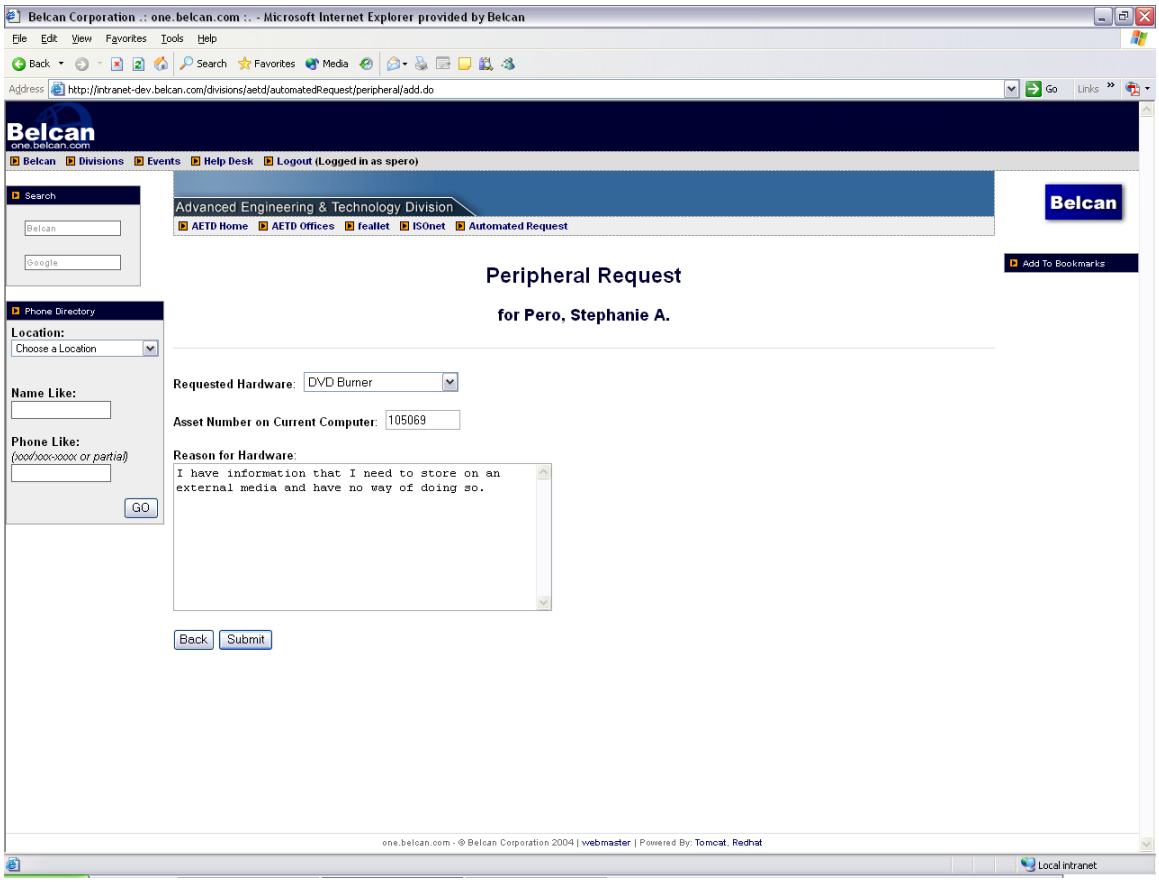


**Figure 50: Filled out Peripheral Request Form**

Figure 51 is an example of what the review page looks like. Notice that the selected option DVD Burner is retained from the previous form.

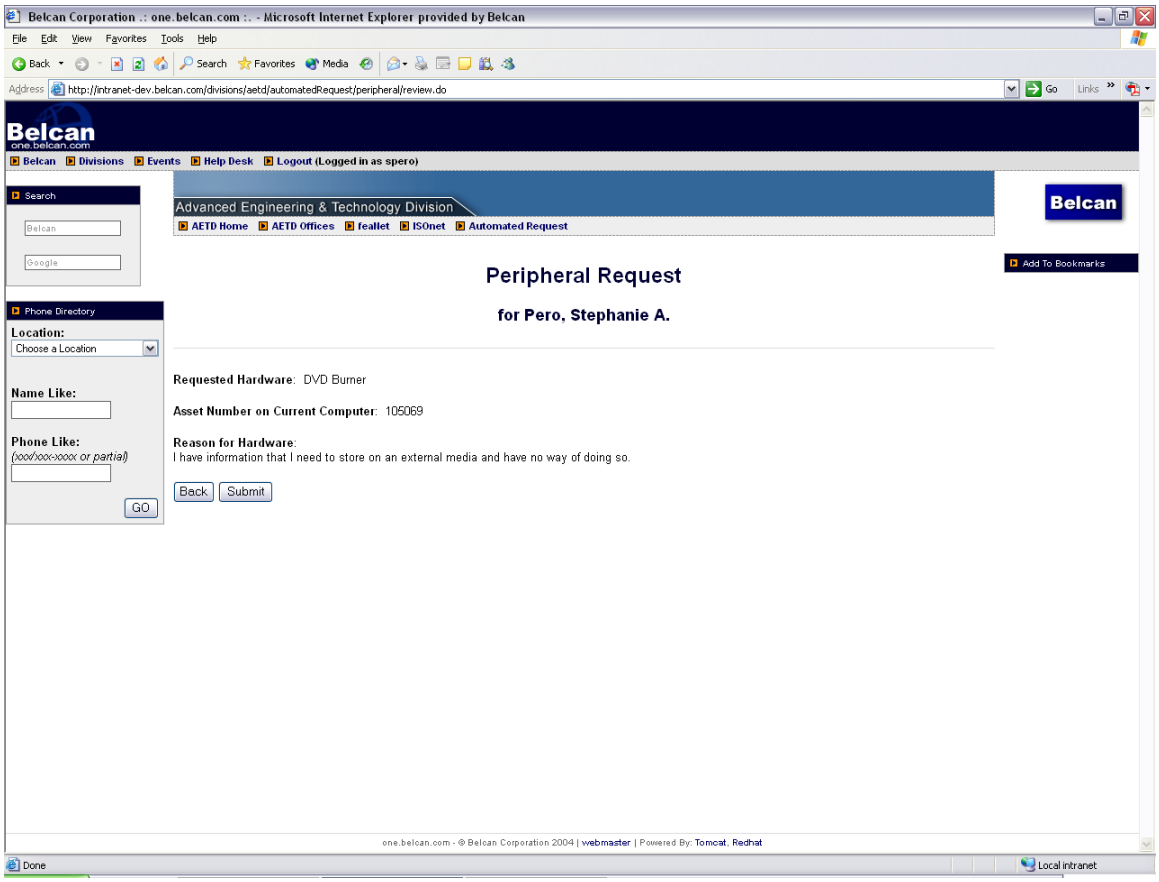


**Figure 51: Peripheral Request Form Review Page**

When the user has decided that this is what they would like to submit for approval they hit the submit button. The submit button will trigger yet another Struts action that will kick off a submittal process. An explanation of the struts.config file that handles all the struts actions is located in Appendix E. Below are a few code snippets of the java class that is responsible for the submittal process. This file is what sends all email notifications to the managers, and inserts or updates all the information into the databases.

First the class checks if the user is logged in with the same code as above which has been fully explained.

```
		if(!CheckUserLogin.isUserLoggedIn(req))
	{
		return mapping.findForward("login");
	}
```

Next, the class creates an instance of the peripheral form which holds all the information that was stored in the beans from the add form. Once the form pf is created a test is ran to check the id of the form and make sure one exists. If the id is a valid id then the answer is checked to see if this request has been approved by anyone and if so then who approved the request. This ensures the proper people get the emails for requests.

```
	PeripheralForm pf = (PeripheralForm)form;
	if(pf.getId() == null || pf.getId().equals("")){
	check = insertRequest(form, req);}
	else if(pf.getAnswer().equals("Accept")){
	if((pf.getGeneralmanager() == null || pf.getGeneralmanager().equals("")) &&
(pf.getOppmanager() == null || pf.getOppmanager().equals("")) && (pf.getItdirector() ==
null || pf.getItdirector().equals("")))
		{
			check = updateRequest(form, req);
		}
		else if((pf.getOppmanager() == null || pf.getOppmanager().equals("")) &&
(pf.getItdirector() == null || pf.getItdirector().equals("")))
		{
			check = updateGM(form, req);
		}
		else if(pf.getItdirector() == null || pf.getItdirector().equals(""))
		{
			check = updateOpp(form, req);
		}
		else if(!pf.getItdirector().equals("") && (pf.getTech() == null ||
pf.getTech().equals("")))
		{
			check = updateDir(form, req);
		}else{
			check = false;
			System.out.println("something is wrong with the approval logic!");
		}
```

```
                }
                else if((pf.getTech() != null && !pf.getTech().equals("")))
                {
                        check = updateTech(form, req);
                }
                else if(pf.getAnswer().equals("Deny"))
                {
                        check = denied(form, req);
                }else{
                        System.out.println("this is broken!");
                        System.out.println(req.getAttribute("tech"));
                }
    if (check)
    {
       return mapping.findForward("success");
    } else {
       return mapping.findForward("failure");
            }
}
```

Once the proper person is selected through the check above it is decided whether the items from the add form need to be inserted or updated. For the purpose of ease and clarity of following along with the process no one has approved the request so an insert will take place and an email will be sent to the direct manager. Below is an example of how an email is generated and how the correct person receives it.

```
//email appropriate people
String emailbody = "The following peripheral request has been submitted for approval by
"+ name[0] + "intended for their Direct Manager.  ";
emailbody += "As the Direct Manager, you can go directly to the review page by clicking
the following link: ";
emailbody += "You can go directly to the review page by clicking the following link:";
emailbody += " <a href=\"http://" + req.getServerName() +
"/divisions/aetd/automatedRequest/peripheral/adminview.do?id=" + id + "\">";
emailbody += "http://" + req.getServerName() +
"/divisions/aetd/automatedRequest/peripheral/adminview.do?id=" + id;
emailbody += "</a>";
String emailsubject = "A new peripheral hardware request has been submitted.";

        HttpSession session = req.getSession();
        User user = (User) session.getAttribute("user");
```

```
        //user's direct manager
        String[] to = new String[1];
        to[0] = ActiveDirectory.getUser(manager).getEmail();
        String from = user.getEmail();
        String[] cc = {"spero@belcan.com"};

//This is sent to the user's Direct Manager, then to the GM, then to OM,
//then IT Dir, then back to Operations Manager, and finally to the appropriate Tech.
//the user will also be notified
        Email.sendToWithCC(to, from, cc, emailsubject, emailbody);
```

This same process is repeated for each approval from the general manager, operations manager, or the IT director. After the user has submitted the request they are redirected to the home page. Figure 52 is an example of this.

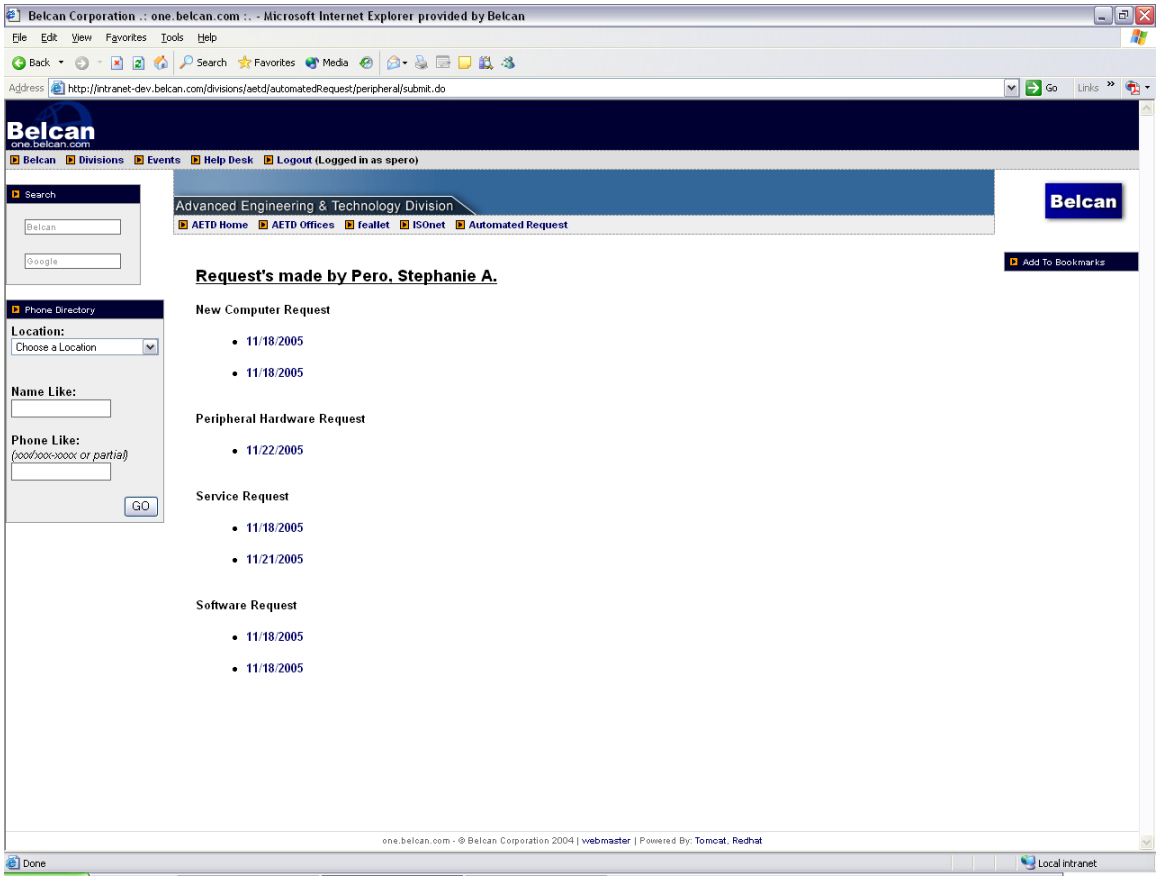


**Figure 52: Automated Request "home" page**

Below is a snapshot of one of the emails that is automatically generated and sent to the appropriate individuals from the java class that is explained above. Notice that on the email it list who the request is for and that it is intended for that person's direct manager. This is to ensure that the receiver on the email is aware of what they are opening.

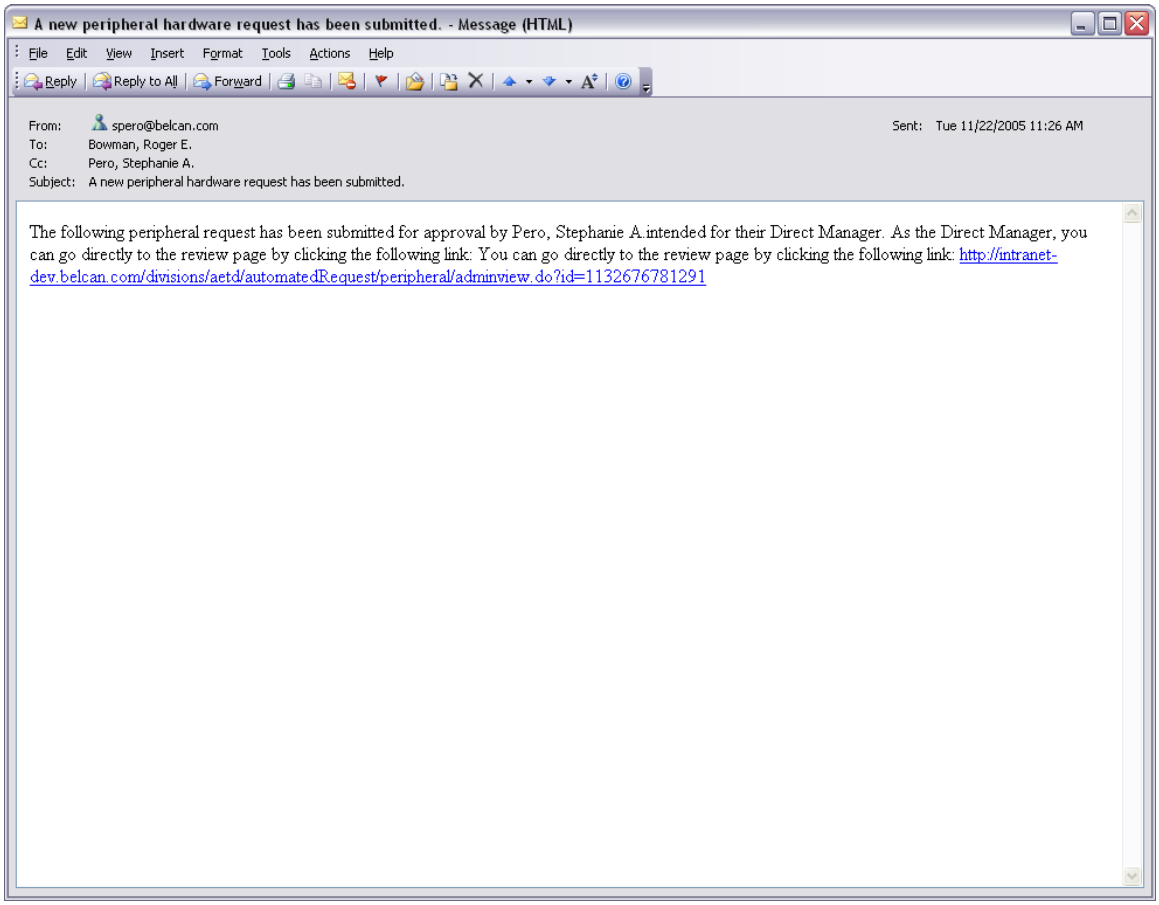


**Figure 53: Email sent to Direct Manager**

When the direct manager follows the link provided in the email, they are brought to this screen shown in figure 54. This is an approval/denial page that allows for the specified manager to either accept or deny.

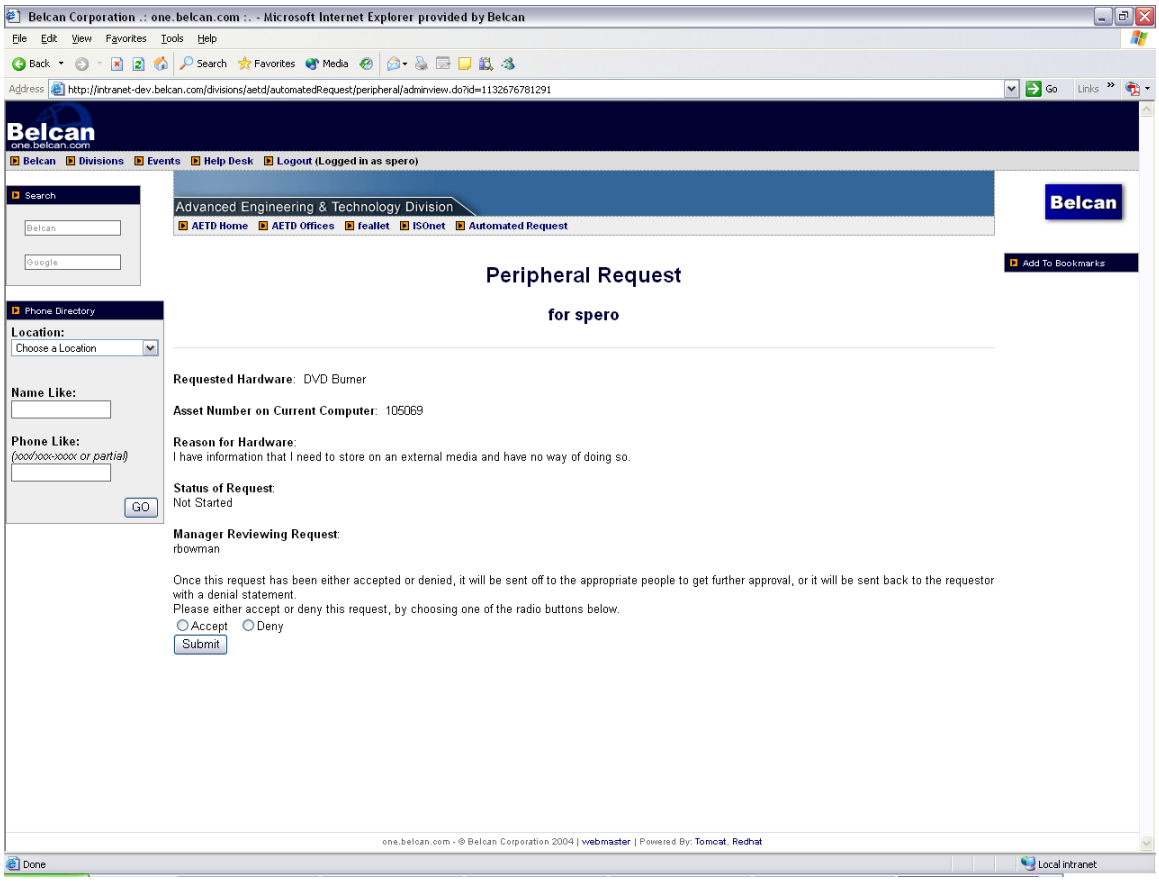


**Figure 54: Peripheral Request Direct Manager's Approval/Denial Screen**

In the form above a java server page is used to display the information. This form like the original add form allows for input and then triggers a struts action when submitted.

<html:form action="/divisions/aetd/automatedRequest/peripheral/submit">

The line of code above is the action that this will perform when the employee hits the submit button. This action accesses the same class as the previous submittal, this one just uses a different function within the class because the direct manager has now accepted the request.

In the java class for this particular step the function updateRequest is used and an update query to the database is done with the answer within this form. For this process I am going to assume the direct manager approves this request. This function also emails an individual to approve the request. This process continues on exactly the same throughout the remainder of the process.

The next step that differs from the rest of the process is when the user accesses his or her status page. This is found by going to home page of the application and clicking on the date of the specified request. Once a user follows the link they are brought to the page below.

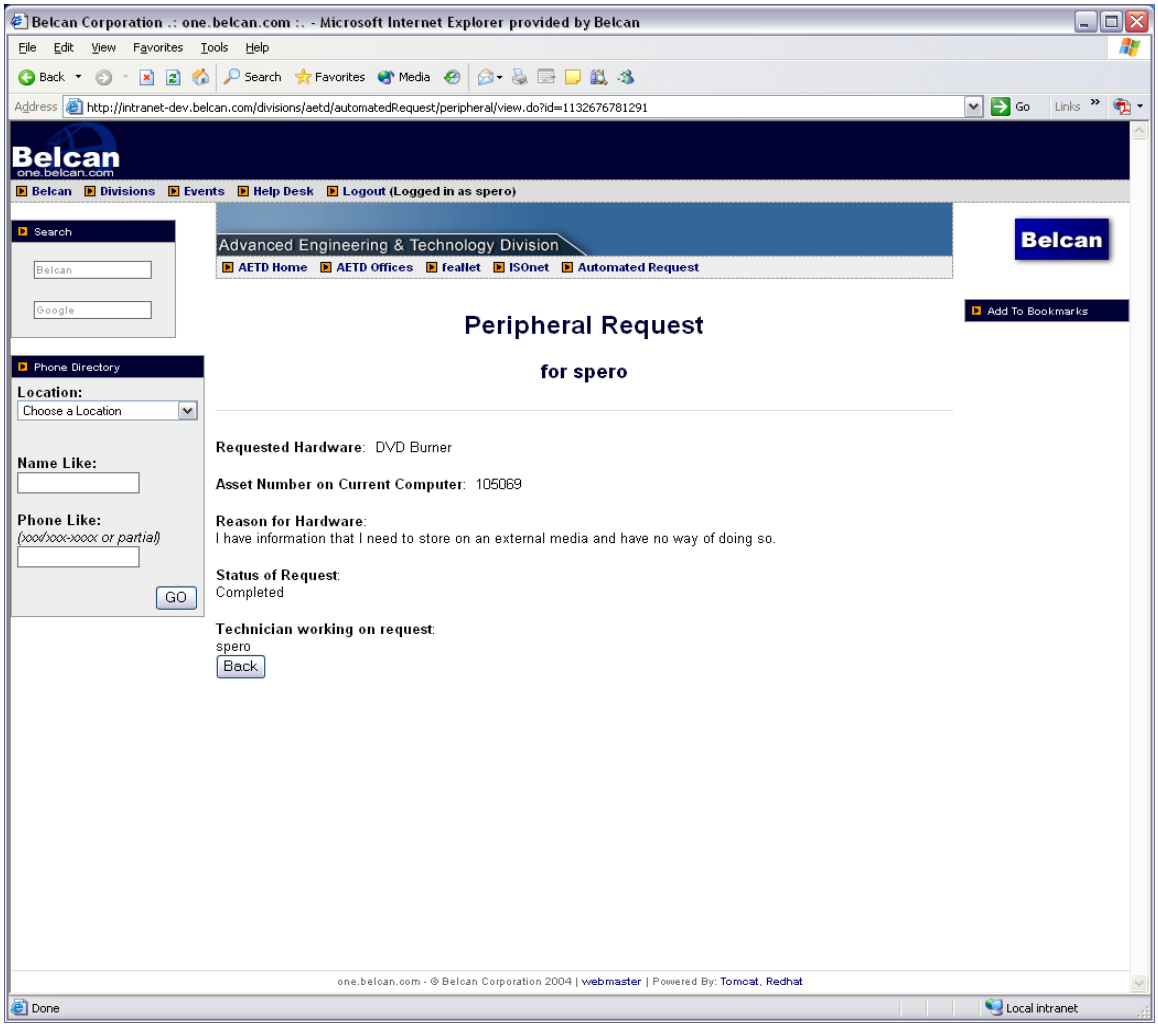


**Figure 55: Status Page for Peripheral Request Form**

In order for an employee to view this information another struts action takes place which calls a view class. The jsp below is what allows for the employee to see the information that is taken from the java file.

Here I am including the admin header which allows for user login.

```
<%@ include file="/includes/adminAetdHeader.jsp" %>
```

Next, I create an instance of a bean named service that is of type service bean which is located in the Beans directory and this allows me to retrieve information that has been submitted from the logged in user.

```
<jsp:useBean id="peripheral" type="Belcan.Beans.Forms.PeripheralBean"
scope="request"/>
```

Then I start the content of this view page by outputting the name 'Service Request' and retrieving and displaying the logged in username via the bean from above.

```
<center>
        <h1>Peripheral Request</h1>
        <h2>for <c:out value="${peripheral.username}"/></h2>
</center>
<hr noshade color="#DDDDDD" size=1>
```

From here on down I am displaying static name or header and then I display the value that is in the database using the bean name from above.

```
        <p>
                <strong>Requested Hardware</strong>: 
                <c:out value="${peripheral.requestedHardware}"/>
        <p>
                <strong>Asset Number on Current Computer</strong>: 
                <c:out value="${peripheral.asset}"/>
        <p>
                <strong>Reason for Hardware</strong>:<br>
                <c:out value="${peripheral.reason}"/>
        <p>
```

```
<p>
        <strong>Status of Request</strong>:<br>
        <c:if test="${peripheral.status == 'Denied'}">
           <c:out value="${peripheral.answer}"/>
           <p>
           <strong>Reason for Denial:</strong>
           <c:out value="${peripheral.denialreason}"/>
        </c:if>
        <c:if test="${peripheral.status != 'Denied'}">
           <c:out value="${peripheral.status}"/>
        </c:if> <br>
```

This is a test to check if a tech has been assigned this request if one has been assigned then it will display that techs name if not this section will be ignored.

```
<c:if test="${!empty peripheral.tech && peripheral.tech != ''}">
        <p>
        <strong>Technician working on request</strong>:<br>
        <c:out value="${peripheral.tech}"/>
</c:if> <br>
```

This is a button that will redirect the user to the page previously viewed.

```
<html:button property="" value="Back" onclick="history.go(-1)"/>
```

And finally this is where I include the footer for closure of the page.

```
<%@ include file="/includes/footer.jsp" %>
```

## Appendix D

### Struts Configuration File

To briefly explain what this file is used for I will take one snippet of this and explain it thoroughly. For this example I will use my service form as the step through example.

First when the user tries to access the form an action takes place that checks if the user is logged in. This is done with the type in the first action statement, type="**Belcan.Util.CheckUserLogin**". The action path for this, path="**/divisions/aetd/automatedRequest/service/add**, represents the url in which the user will see when going to the specified add page. When the user is logged in the return mapping within the class file returns a success, name="**success**", so it automatically redirects the user to the specified path in this action, path="**/WEB-INF/jsp/service/service.jsp**".

Once the user has submitted the form for the first time another action is triggered forwarding the user to the review page, path="**/divisions/aetd/automatedRequest/service/review**", the action input is the path of the actual jsp that the path is going to point at, input="**/WEB-INF/jsp/service/service.jsp**". The information from the add form is put into a form bean which is the place holder for all the session information, this is called and utilized by the statement, name="**ServiceForm**". The review page that the user is forwarded to has a precondition that the user must be logged in which is called by the type="**Belcan.Util.CheckUserLogin**". If this is true then the user is then forwarded to the desired location, forward name="**success**" path="**/WEB-INF/jsp/service/review_service.jsp**".

Once the user has reviewed the information they will then submit it to be approved. The action that is triggered in the review page is the submit function. Before this submit gets called the form must retrieve all the input information from the form bean ServiceForm, and this information is grabbed by action input="**/WEB-INF/jsp/service/review_service.jsp**" name="**ServiceForm**". Once the action is triggered the user id redirected to path="**/divisions/aetd/automatedRequest/service/submit**" and the type="**Belcan.Forms.Service.ServiceSubmit**" ensures that the information from the bean gets forwarded to the ServiceSubmit.java file and the user will get redirected to the index page by, path="**/divisions/aetd/automatedRequest/index.do**".

Once the user submits a request the same actions take place for the entire approval/denial process all the way to the closeout of the request. The next action to be triggered is the viewing of an item. When a user wants to view something an action is triggered by, path="**/divisions/aetd/automatedRequest/service/view**". This information is retrieved using the ServiceView.java class, type="**Belcan.Forms.Service.ServiceView**"> and then forwarding, path="**/WEB-INF/jsp/service/view_service.jsp**" the information to the view page for the user. This is true for each user including employees, managers, and technicians.

This is a complete explanation of how this systems works behind the scenes using struts. Below is the file that I use in my application to make everything work seamlessly.

Full struts.config file:

```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
  <!DOCTYPE struts-config (View Source for full doctype...)>
<struts-config>
<data-sources />
<form-beans>
<form-bean name="PeripheralForm"
type="Belcan.Forms.Peripheral.PeripheralForm" />
<form-bean name="NewComputerForm"
type="Belcan.Forms.NewComputer.NewComputerForm" />
<form-bean name="ServiceForm" type="Belcan.Forms.Service.ServiceForm"
/>
<form-bean name="SoftwareForm"
type="Belcan.Forms.Software.SoftwareForm" />
</form-beans>
<global-exceptions />
<global-forwards>
<forward name="welcome" path="/Welcome.do" />
<forward name="login" path="/login/" redirect="true" />
</global-forwards>
<action-mappings>
<action path="/divisions/aetd/automatedRequest/tech/index"
type="Belcan.TechIndex" scope="request">
<forward name="success" path="/WEB-INF/jsp/forms/fix_view/index.jsp"
/>
<forward name="failure" path="/index.jsp" />
</action>
<action path="/divisions/aetd/automatedRequest/service/add"
scope="request" type="Belcan.Util.CheckUserLogin">
<forward name="success" path="/WEB-INF/jsp/service/service.jsp" />
</action>
<action input="/WEB-INF/jsp/service/service.jsp" name="ServiceForm"
path="/divisions/aetd/automatedRequest/service/review" scope="request"
type="Belcan.Util.CheckUserLogin" validate="true">
<forward name="success" path="/WEB-INF/jsp/service/review_service.jsp"
/>
</action>
<action path="/divisions/aetd/automatedRequest/service/view"
scope="request" type="Belcan.Forms.Service.ServiceView">
<forward name="success" path="/WEB-INF/jsp/service/view_service.jsp" />
</action>
<action path="/divisions/aetd/automatedRequest/service/adminview"
scope="request" type="Belcan.Forms.Service.ServiceView">
<forward name="success" path="/WEB-
INF/jsp/admin/adminview_service.jsp" />
</action>
<action path="/divisions/aetd/automatedRequest/service/techview"
scope="request" type="Belcan.Forms.Service.ServiceView">
<forward name="success" path="/WEB-
INF/jsp/admin/tech/techview_service.jsp" />
</action>
```

```
<action input="/WEB-INF/jsp/service/review_service.jsp"
name="ServiceForm"
path="/divisions/aetd/automatedRequest/service/submit" scope="request"
type="Belcan.Forms.Service.ServiceSubmit">
<forward name="success"
path="/divisions/aetd/automatedRequest/index.do" />
</action>

<action path="/divisions/aetd/automatedRequest/software/add"
scope="request" type="Belcan.Util.CheckUserLogin">
<forward name="success" path="/WEB-INF/jsp/software/software.jsp" />
</action>
<action input="/WEB-INF/jsp/software/software.jsp" name="SoftwareForm"
path="/divisions/aetd/automatedRequest/software/review"
scope="request" type="Belcan.Util.CheckUserLogin" validate="true">
<forward name="success" path="/WEB-
INF/jsp/software/review_software.jsp" />
</action>
<action path="/divisions/aetd/automatedRequest/software/view"
scope="request" type="Belcan.Forms.Software.SoftwareView">
<forward name="success" path="/WEB-INF/jsp/software/view_software.jsp"
/>
</action>
<action path="/divisions/aetd/automatedRequest/software/adminview"
scope="request" type="Belcan.Forms.Software.SoftwareView">
<forward name="success" path="/WEB-
INF/jsp/admin/adminview_software.jsp" />
</action>
<action path="/divisions/aetd/automatedRequest/software/techview"
scope="request" type="Belcan.Forms.Software.SoftwareView">
<forward name="success" path="/WEB-
INF/jsp/admin/tech/techview_software.jsp" />
</action>
<action input="/WEB-INF/jsp/software/software.jsp" name="SoftwareForm"
path="/divisions/aetd/automatedRequest/software/submit"
scope="request" type="Belcan.Forms.Software.SoftwareSubmit">
<forward name="success"
path="/divisions/aetd/automatedRequest/index.do" />
</action>
<action path="/divisions/aetd/automatedRequest/newComputer/add"
scope="request" type="Belcan.Util.CheckUserLogin">
<forward name="success" path="/WEB-
INF/jsp/new_computer/newComputerRequest.jsp" />
</action>
<action input="/WEB-INF/jsp/new_computer/newComputerRequest.jsp"
name="NewComputerForm"
path="/divisions/aetd/automatedRequest/newComputer/review"
scope="request" type="Belcan.Util.CheckUserLogin" validate="true">
<forward name="success" path="/WEB-
INF/jsp/new_computer/review_new.jsp" />
</action>
<action path="/divisions/aetd/automatedRequest/newComputer/view"
scope="request" type="Belcan.Forms.NewComputer.NewComputerView">
```

```
<forward name="success" path="/WEB-
INF/jsp/new_computer/view_new.jsp" />
</action>
<action
path="/divisions/aetd/automatedRequest/newComputer/adminview"
scope="request" type="Belcan.Forms.NewComputer.NewComputerView">
<forward name="success" path="/WEB-INF/jsp/admin/adminview_new.jsp"
/>
</action>
<action path="/divisions/aetd/automatedRequest/newComputer/techview"
scope="request" type="Belcan.Forms.NewComputer.NewComputerView">
<forward name="success" path="/WEB-
INF/jsp/admin/tech/techview_new.jsp" />
</action>
<action input="/WEB-INF/jsp/new_computer/review_new.jsp"
name="NewComputerForm"
path="/divisions/aetd/automatedRequest/newComputer/submit"
scope="request" type="Belcan.Forms.NewComputer.NewComputerSubmit">
<forward name="success"
path="/divisions/aetd/automatedRequest/index.do" />
</action>
<action path="/divisions/aetd/automatedRequest/peripheral/add"
scope="request" type="Belcan.Util.CheckUserLogin">
<forward name="success" path="/WEB-INF/jsp/peripheral/peripheral.jsp" />
</action>
<action input="/WEB-INF/jsp/peripheral/peripheral.jsp"
name="PeripheralForm"
path="/divisions/aetd/automatedRequest/peripheral/review"
scope="request" type="Belcan.Util.CheckUserLogin" validate="true">
<forward name="success" path="/WEB-
INF/jsp/peripheral/review_peripheral.jsp" />
</action>
<action path="/divisions/aetd/automatedRequest/peripheral/view"
scope="request" type="Belcan.Forms.Peripheral.PeripheralView">
<forward name="success" path="/WEB-
INF/jsp/peripheral/view_peripheral.jsp" />
</action>
<action path="/divisions/aetd/automatedRequest/peripheral/adminview"
scope="request" type="Belcan.Forms.Peripheral.PeripheralView">
<forward name="success" path="/WEB-
INF/jsp/admin/adminview_peripheral.jsp" />
</action>
<action path="/divisions/aetd/automatedRequest/peripheral/techview"
scope="request" type="Belcan.Forms.Peripheral.PeripheralView">
<forward name="success" path="/WEB-
INF/jsp/admin/tech/techview_peripheral.jsp" />
</action>
<action input="/WEB-INF/jsp/peripheral/review_peripheral.jsp"
name="PeripheralForm"
path="/divisions/aetd/automatedRequest/peripheral/submit"
scope="request" type="Belcan.Forms.Peripheral.PeripheralSubmit">
<forward name="success"
path="/divisions/aetd/automatedRequest/index.do" />
```

```
</action>
</plug-in>
</struts-config>
```

# References

1.) Berberich, Donald. "Re: Request for Project" E-mail to Stephanie Pero. April 12, 2005.

2.) Berberich, Donald. S.O.W. May 4, 2005.

3.) Bowman, Roger. Web Administrator, Belcan Corporation. Personal Interview May 3, 2005.

4.) Crosby, Scott. UNIX Admin, Belcan Corporation. Personal Interview April 29, 2005.

5.) Faris, Earl. Project Manager, Belcan Corporation. Personal Meeting/Interview May11, 2005.

6.) Perazzo, Richard. Operations Manager, Belcan Corporation. Personal Meeting/Interview May 11, 2005.

7.) http://www.belcan.com/company/about_belcan.htm Copyright 2001. Belcan Corporation. May 13, 2005.

8.) http://www.infrastructures.org/ Copyright 1994-2004 Steve Traugott, Joel Huddleston, Joyce Cao Traugott. May 13, 2005.

9.) http://support.dell.com/ Copyright 1999-2005 Dell Inc. May 13, 2004

10.) http://www.westgatemedia.com/products/adobe_phs_7.php Copyright © 2001 - 2004 Westgate Media Solutions. May16, 2005

11.) http://www.edirectsoftware.com/default/Dreamweaver.html Copyright © 2001-2005 eDirectSoftware.com. May16, 2005

12.) http://www.novell.com/products/linuxenterpriseserver/pricing.html Copyright © 2005 Novell Inc. May16, 2005