LCS Content Document Application

Jake Hochstadt

KENNEDY SPACE CENTER

Major: Computer Engineering

USRP Spring Session

Date: 11 MAY 11

NASA USRP Internship Final Report

LCS Content Document Application

Jake Hochstadt
University of Central Florida, Orlando, Florida, 32816

Abstract

My project at KSC during my spring 2011 internship was to develop a Ruby on Rails application to manage Content Documents. A Content Document is a collection of documents and information that describes what software is installed on a Launch Control System Computer. It's important for us to make sure the tools we use everyday are secure, up-to-date, and properly licensed. Previously, keeping track of the information was done by Excel and Word files between different personnel. The goal of the new application is to be able to manage and access the Content Documents through a single database backed web application. Our LCS team will benefit greatly with this app. Admin's will be able to login securely to keep track and update the software installed on each computer in a timely manner. We also included exportability such as attaching additional documents that can be downloaded from the web application. The finished application will ease the process of managing Content Documents while streamlining the procedure. Ruby on Rails is a very powerful programming language and I am grateful to have the opportunity to build this application.

Nomenclature

LCS = Launch Control Systems

SET = Operation or development room holding hardware and software used to monitor and control end items

LDS = Launch Control System Development Set

FR1 = Fire Room 1

ADS = Application Development Set

FRD = Firing Room Demo Set

KSC = Kennedy Space Center

RoR = Ruby on Rails

OSI = Operating System Image

NDC = NASA Domain Control

CD = Content Document

I. Introduction

I Began my experience at KSC with very little knowledge and background with Ruby on Rails. My only work experience was working part time at RadioShack and with a webhosting company called Hostdime. When I received a phone call from Benita Desuza asking me if I was still interested in an internship at NASA, I was almost speechless. I quickly accepted the internship without any hesitation. With the very little experience I had with RoR, I knew I had to impress and learn this new language quickly.

The project chosen for me was to develop a Rails application for the LCS team to help organize the OSI documents. I worked very closely with another intern, Ricardo Muniz Rodriguez. Ricardo is a Computer Science student from the University of Puerto Rico. We worked together, using our ideas and teamwork, to develop and deploy the application.

II. Ruby on Rails

Ruby on Rails is an open source web application framework. Yukihiro Matsumoto, a Japanese computer scientist, developed the Ruby programming language in 1995. David Hansson, a Danish programmer, built the original framework for Rails in 2003. Rails has a very vibrant ecosystem with more than 1800 contributors helping make the language better with each version. Many of today's websites are also powered by Rails such as Twitter, Groupon, Basecamp, Hulu, and Yellow Pages.com. We used Rails 3 and Ruby 192 to build the LCS application. More information can be found at rubyonrails.org.

III. The LCS Project

When the user first visits the application, they're prompted to enter a username and password. Security was a primary concern when we developed the application. We did not want any NASA employee to be able to view high

risk information such as IP addresses and serial numbers. We were able to incorporate the NASA standard (NDC) login credentials to authenticate users attempting to enter the system.. User authorization was handled by a local database. We chose this way of authenticating because it improves the user experience while also benefitting from the best practices of user management through the NDC single-sign-on service. Also, not all NDC personnel can login and use the application. If the correct credentials are provided for the NDC account, the user will also need to be on the local database to view data.

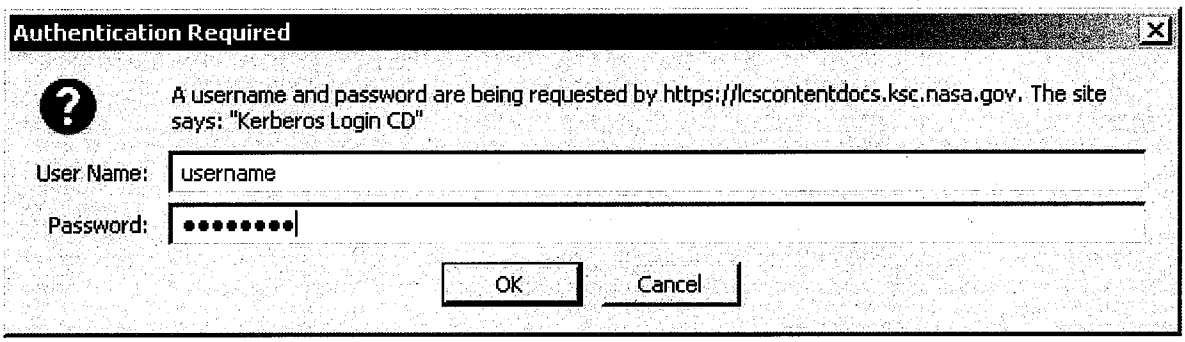


Figure 1 Authentication

Once logged in, the user will be able to see the login status at the bottom right. We setup different permissions depending on the user's role on the local database. The roles are: Root, Admin and User. Each custom privilege allows for certain items in the database to be viewable and/or editable. The entire layout can be seen in figure 4.

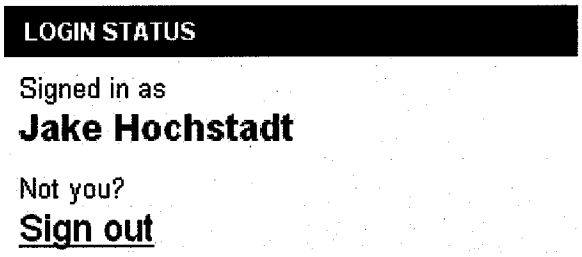


Figure 2 Login Status

For initial successful login, there will be a blue information popup toward the right of the application. This was a required prompt that can also be seen in Figure 4. Once the user navigates throughout the application though, they won't be prompted again. But if the user attempts to view data they're not given permissions to do so, they will be prompted with a red warning message and be redirected to the default home page of the application.

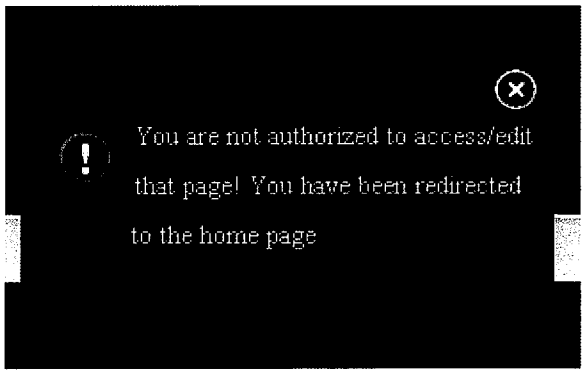
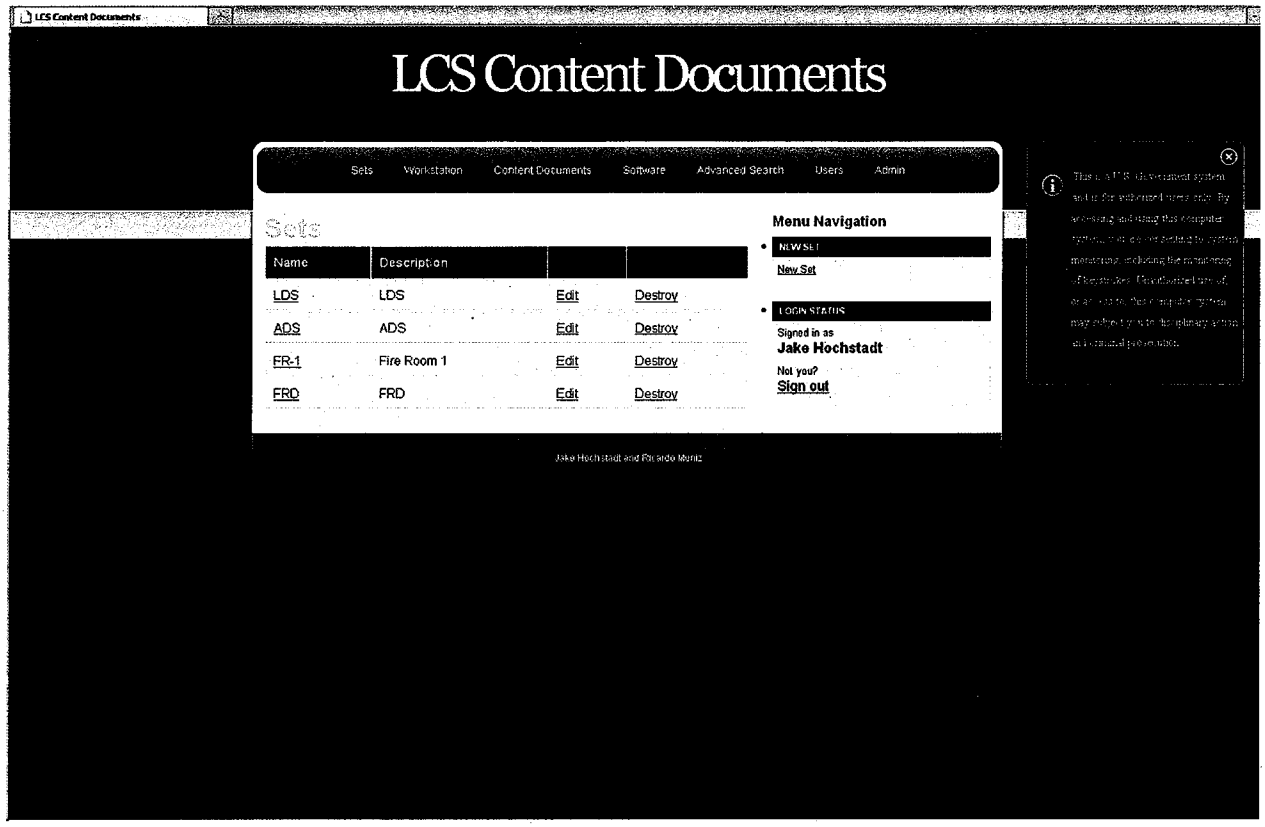


Figure 3 Red Warning Message

Figure 4 Layout



Throughout the application, the user will be able to view, edit or destroy content in the database. It's organized through tabs viewable at the top. Validations are incorporated throughout to remove nil errors in the database and ensure that all data is valid. For instance, if a user tries to create a new Set without a name or description, they will be prompted with an error.

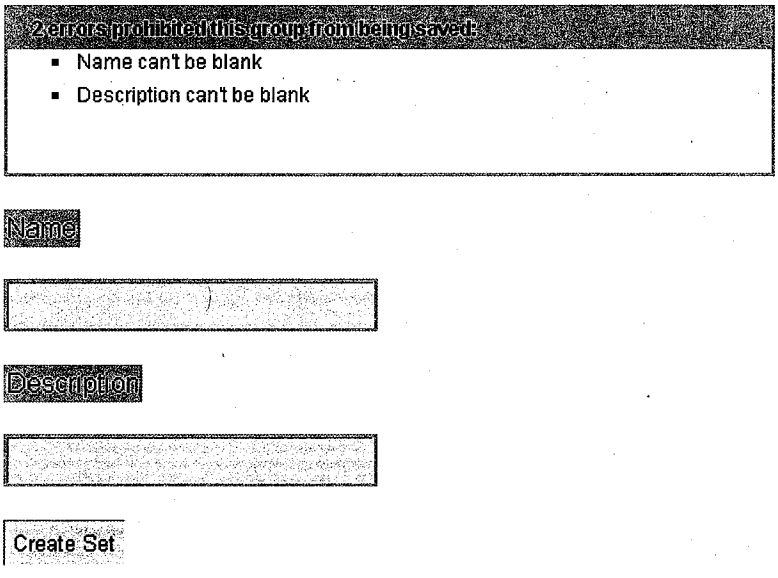


Figure 5 New Set Validations

Creating a new content document gives the user many additional options. They have the ability to add multiple software instances to a single CD. They can also attach documents, such as a Word document, containing additional information about the CD and retrieve the document from the server at a later time. There are also predefined templates for the Name, Image, Test Document, Generation Document and Security Report. Once the user selects the Name or Image, the "###" will disappear. Furthermore, when creating a new CD, the user may not have the Test Document, Generation document or Security report at the time. We prefilled N/A in the boxes, but if the user clicks in the box, it disappears and allows them to type the information in. We tried to ease the process for creating a new CD.

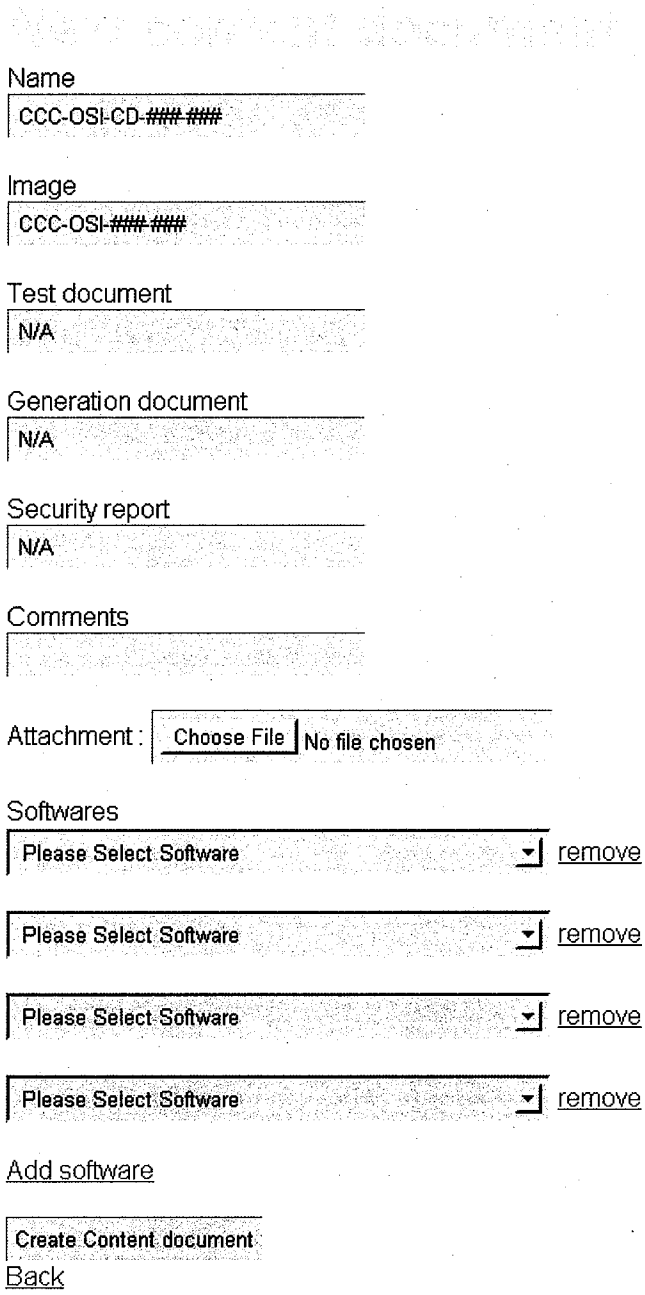


Figure 6 New Content Document

Workstations

Name	Location	Serial Number	IP Address		
hostname1	SSPF 11111	LCS1111-1111111	111.111.1.1	Edit	Destroy
hostname10	SSPF 2222	LCS2222-2222222	222.222.222.22	Edit	Destroy

Figure 7 Workstation

Name	Location		
hostname1	SSPF 11111	Edit	Destroy
hostname10	SSPF 2222	Edit	Destroy

Figure 8 Workstation

We customized the view of the workstations page depending on what type of permissions the user has been given. In figure 7, the workstation is viewed logged in as root, which can view/edit the entire database. In figure 8, the workstation is logged in as a user, which can only view the Location and Name, as per our client's request. They will also be redirected to the home page with the red error message if they attempt to edit or destroy information.

The advanced search tab allows the user to quickly search for specific content documents that contain the software added to the search. In the example in figure 9, the user searched for a CD containing PowerPoint, Word and Excel. From the results, there is only 1 CD that contains all three software. We've hyperlinked the CD so the user can quickly view more information.

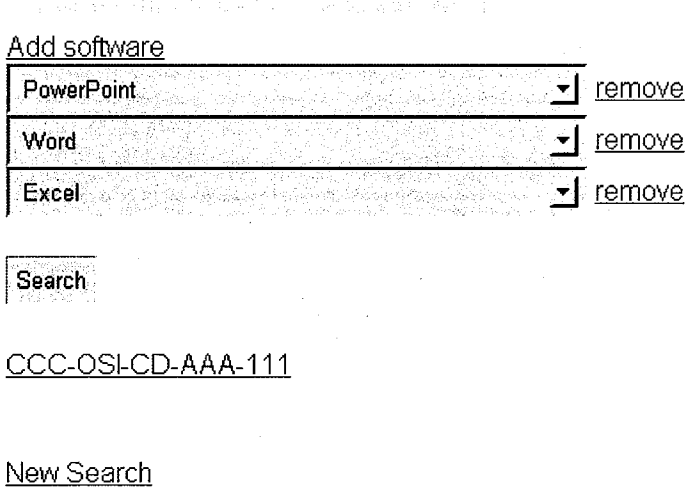


Figure 9 Advanced Search

A. Kerberos

Toward the latter half of my internship, I began working more on server side projects. One of the requests Adam had for our application was to secure it with an SSL and have an Active Directory authentication. I worked closely