Kevin Ireland

CSC 440

JACKN-Wiki

User Roles: Design Manual

## **Problem definition**

In any system, some information or actions are more sensitive than others. Often times this problem is addressed by giving users a certain set of permissions which allow them to access the items of the system that they should have access to. In our system, users can perform any action which could get out of hand as we get more users. We needed to limit people using the stronger actions like deleting, as well as only letting some people do things like editing and moving pages.

## **Requirements (User Stories)**

When a user is first created they will not have any roles assigned to them. At this point they are able to navigate the site and create new pages. An admin can access the Roles page to assign a user any role they see fit. Currently there are two roles that are coded into the system, moderator and admin. If a user is given the role of moderator, they can then edit and move pages. If a user is given the role of admin, they can do anything on the wiki, including specifically deleting pages and accessing the Roles page. If a user tries to take one of these actions without the specified role, they will be brought to a screen telling them that they do not have permission to perform their current action and if they feel they should, to contact an admin.

## **Design**

My feature is designed in a RESTful API style way. It uses the Flask decorator to create a route '/roles' which is a Python function in routes.py called roles(). This function is called when the user routes to the specified URL.

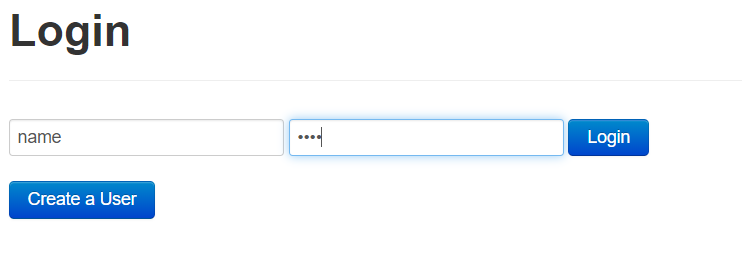
On that template, all the current users of the system are displayed along with their current set of roles. This is all parsed from the users.json file. There is also a button “Add Role” that will take you to the route ‘/add\_role’ where a form “AddRoleForm” is called.

In the add\_role route, there are two textfields, ‘name’ and ‘role’, and a confirmation button, ‘Add Role!’. When these are filled out and the button is pushed, the form returns these two pieces of data where they are used to find a user by the name, and then add the role to that user using the new function defined in user.py, ‘add\_role(role)’.

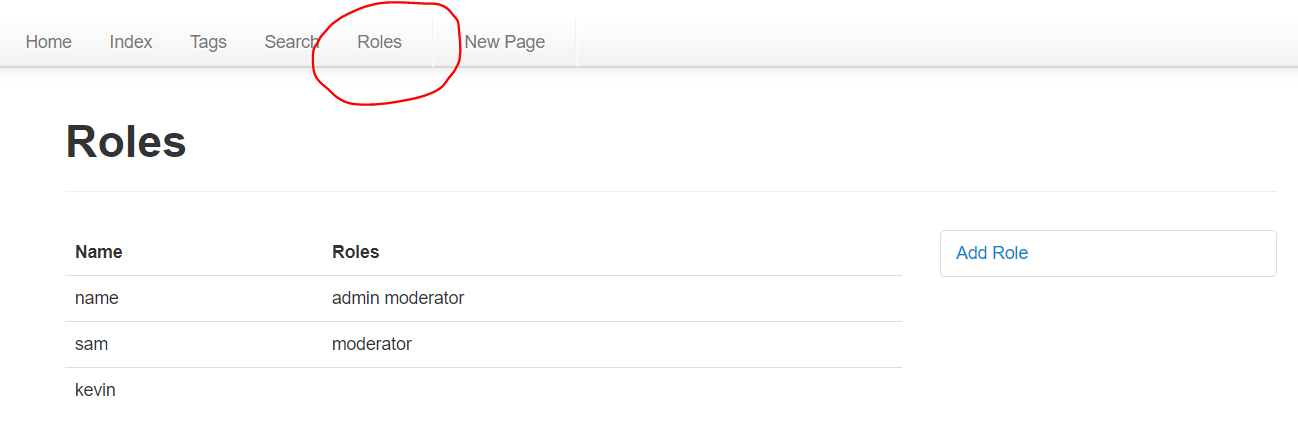
I added two other methods to user.py, is\_moderator and is\_admin, which will check the current user’s roles to see if the role is present. These methods are called in several of the routes to check the current user’s role to see if they should have permission to access the action they are trying to perform. If they do not, they are redirected to a template ‘denied.html’ that tells them they do not have access.

User Roles: User Manual

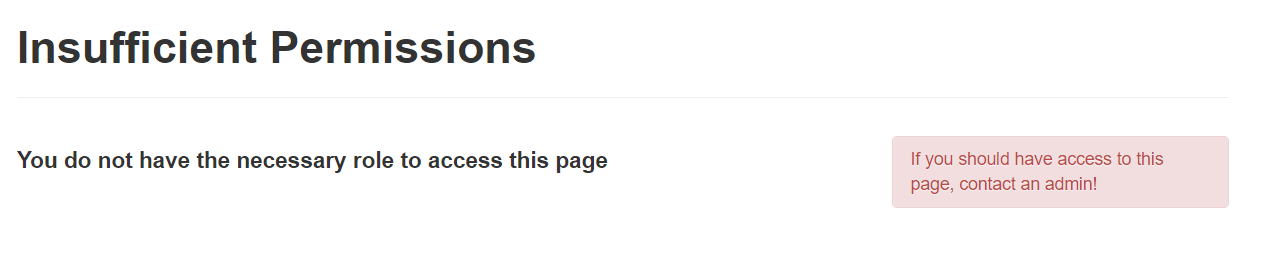
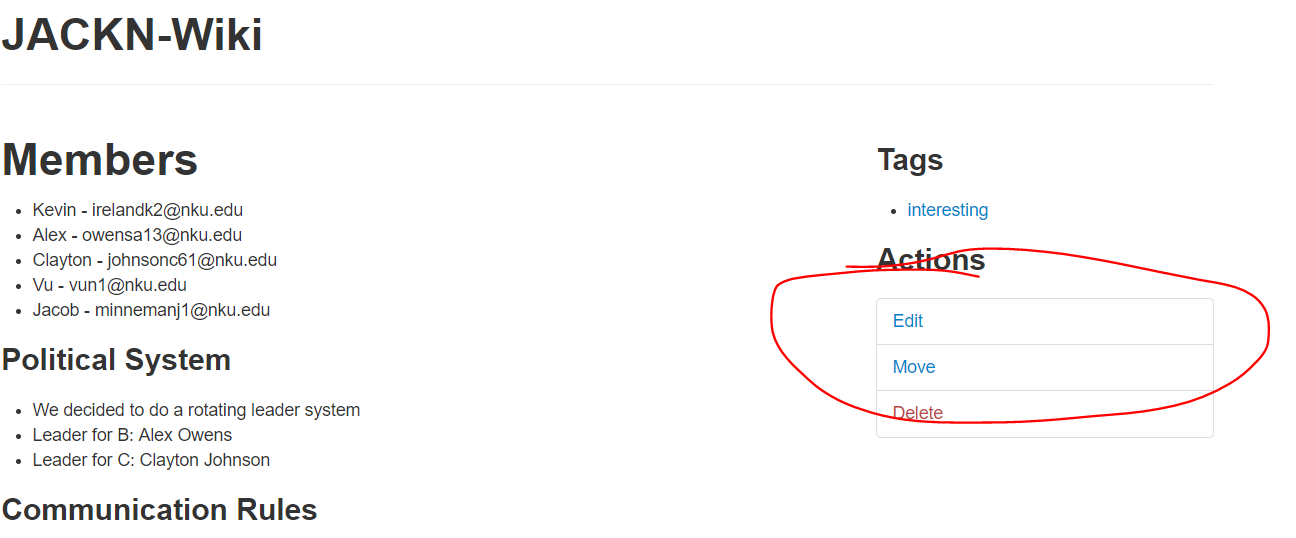
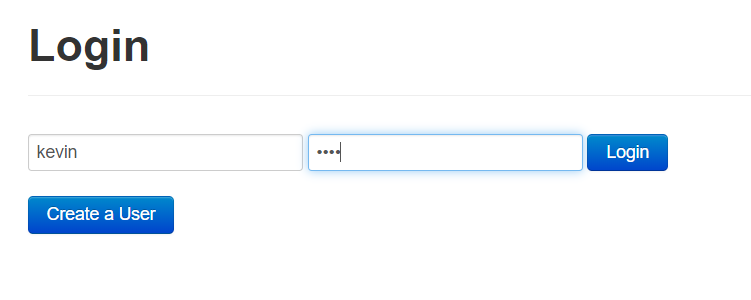
Step 1: Login to your user.



Step 2: Navigate to the roles page to look at the current system’s users and their roles



Step 3: Now we can see that the user “kevin” does not have any roles, let’s try logging in as that user and accessing the “edit” action



Step 4: Let’s go back to our first user and add the moderator role to “kevin”

