Game Contest Server

User Administration

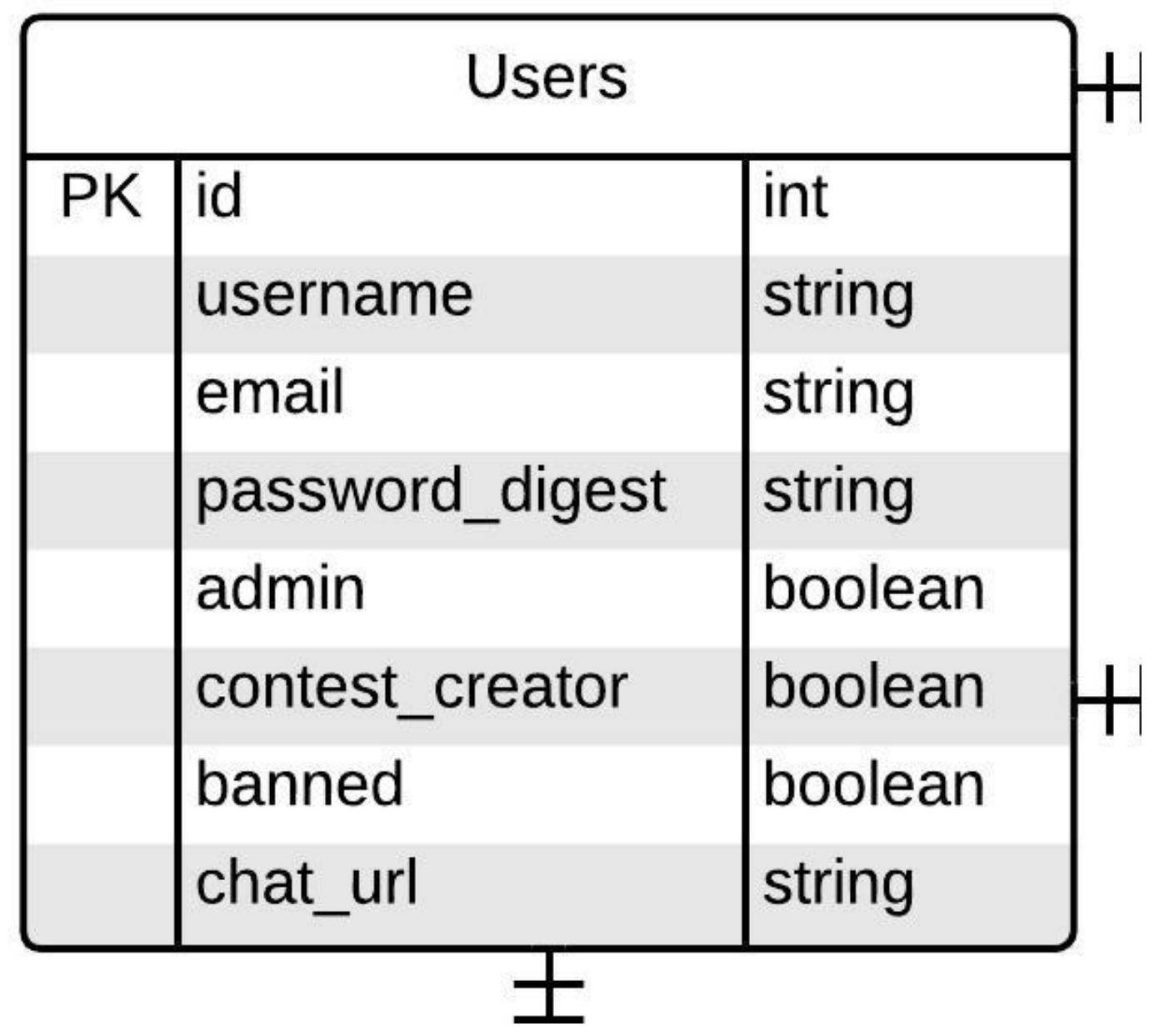
Nathan Lundell
Department of Computer Science and Engineering, Taylor University, Upland, Indiana 46989

Introduction

The "Game Contest Server" is a web application developed using the Ruby on Rails framework, which is a web-programming framework using the popular Ruby language. The "Game Contest Server" was created to host class game project contests so that student-made players could be easily tested against each other in a variety of different contest styles. "Game Contest Server" was created to be incredibly flexible, being both game and programming language agnostic.

What Is An Administrator?

Every website needs a person or group who has the ability to take control of other users when necessary and to give or take away special privileges a user might have on a website. The "Game Contest Server" will host many matches, contests, and tournaments between many different users. While some users will be allowed to schedule these events, most users are restricted to only being able upload their player code to the server to be run against others during these events. The administrator(s) have a level of control above either of these user types.



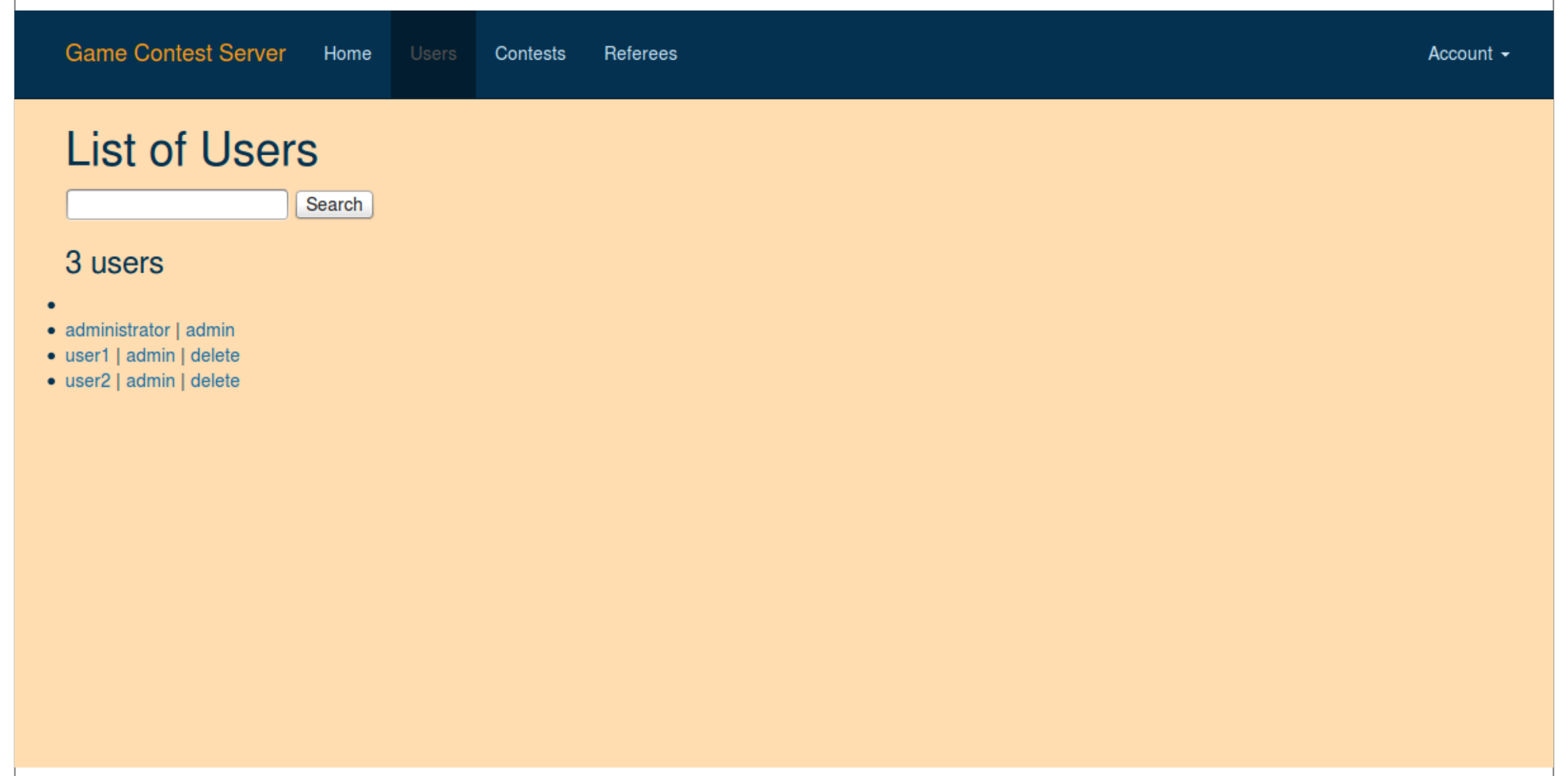
Users		
PK	id	int
	username	string
	email	string
	password_digest	string
	admin	boolean
	contest_creator	boolean
	banned	boolean
	chat_url	string

Besides their basic information of username, email, chat identification and password, the user has three other important flags that are stored in the database. As boolean variables are True/False flags, they can be used to turn user functionality on and off just by switching them on or off, true or false. Only a user with the admin boolean set to "true" is allowed to change these three flags for either themselves or for other users.

Dashboard vs. Popup

The original plan for integrating administration tools into the website was to create a dashboard that could be accessed through a link on the navigation bar. However, this presented an obstacle when wanting to easily administrate multiple users. It would have limited searching to one individual user at a time. In addition, it would have been a bad use of page space, using an entire new page for three checkboxes.

Instead it was decided to integrate the administration tools into the user search area, and to insert links to a admin popup. This allows for easy administration of multiple users as it does not require searching for a single user, but can operate easily with a list of users. In the future, this will be very helpful when wanting to administer multiple users that are a part of a certain group. This is also a better use of space as the admin options are limited to three options: banned, contest creator, and admin.

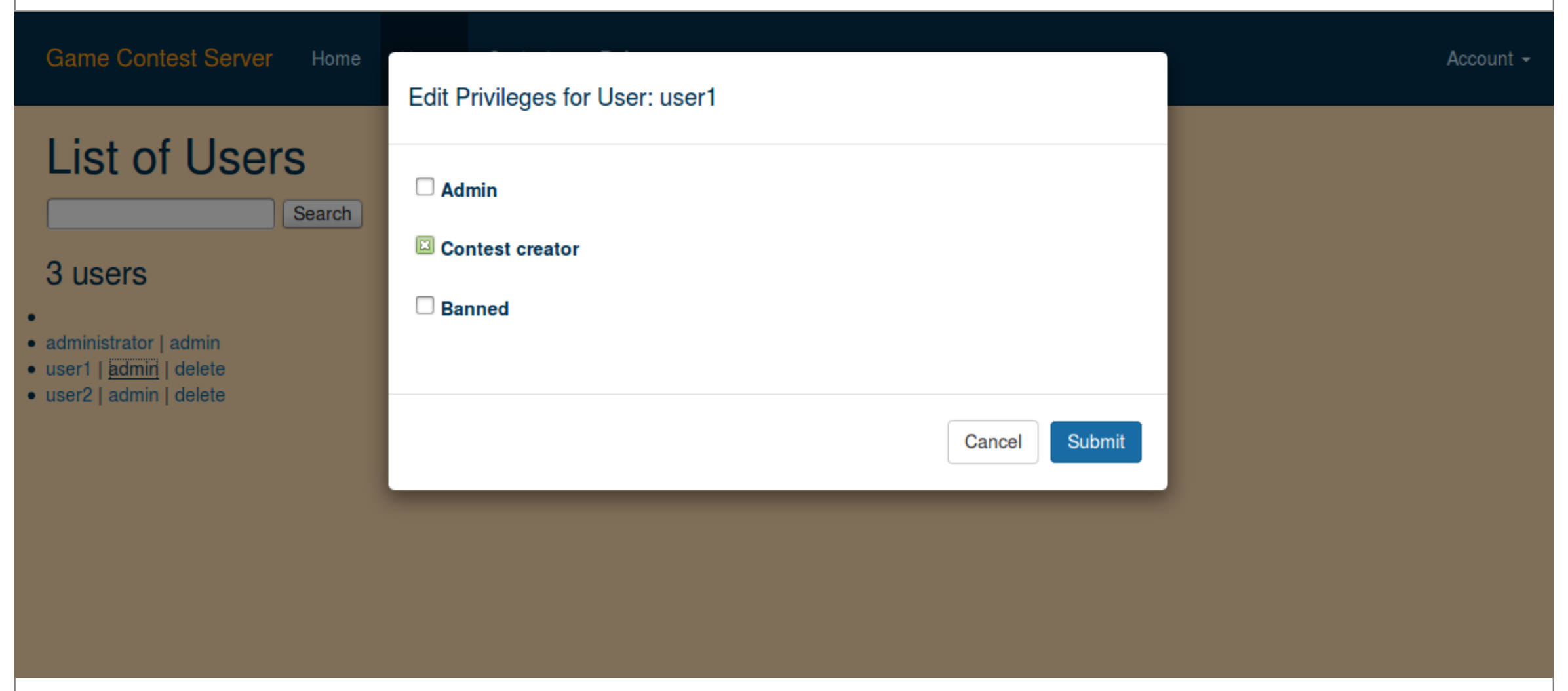


List of Users

3 users

-
- administrator | admin
- user1 | admin | delete
- user2 | admin | delete

Currently, when the admin is logged in, two links show up next to a user. "Delete" deletes the user on the same line as the link. A rule was created that does not allow the user to delete themselves, which is why is does not show next to the administrator in the screenshot above. An administrator can delete anyone but themselves. Clicking on the "admin" link brings up the popup.



List of Users

3 users

-
- administrator | admin
- user1 | admin | delete
- user2 | admin | delete

Edit Privileges for User: user1

☐ Admin

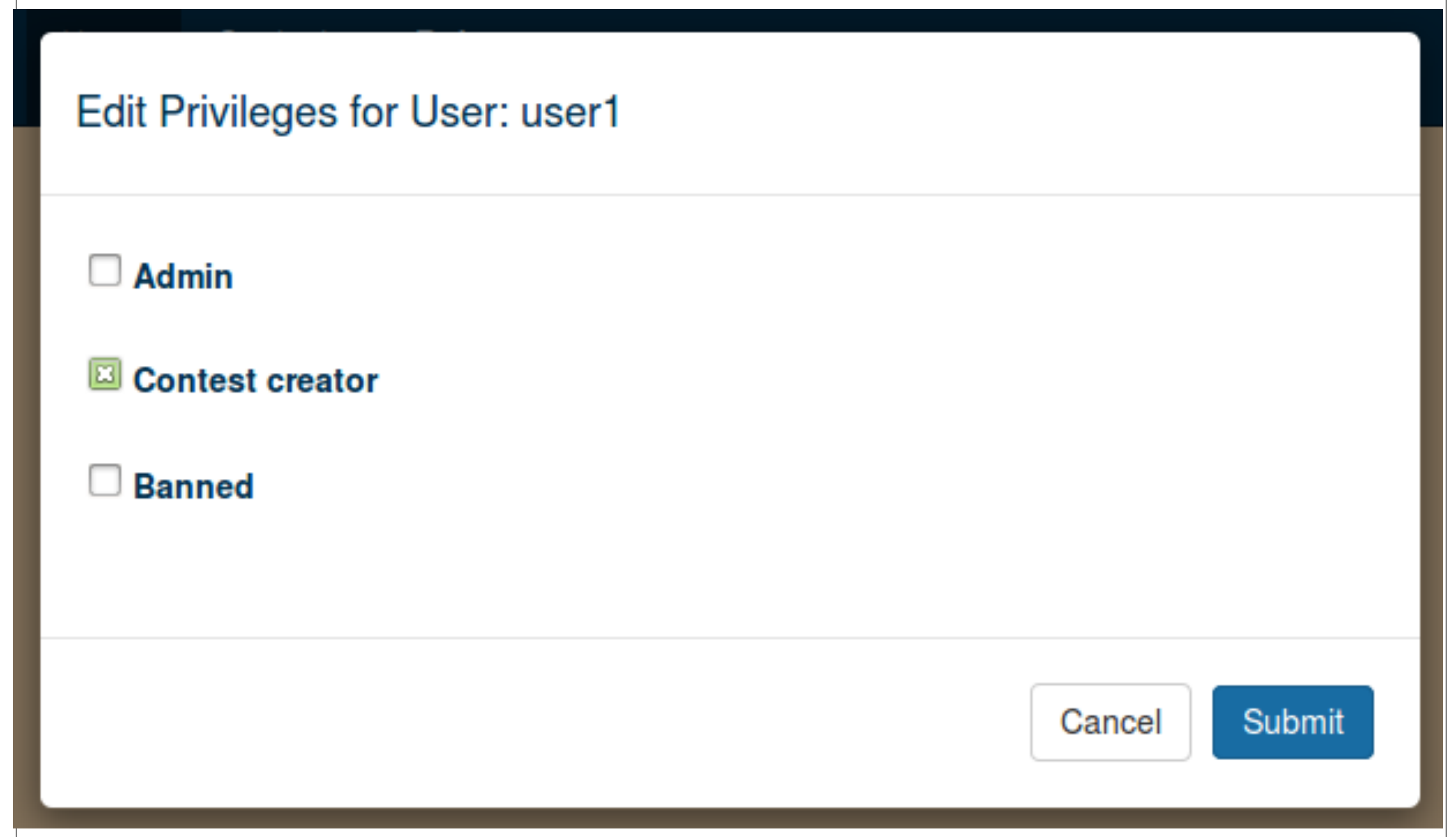
☒ Contest creator

☐ Banned

How It Works

The admin popup is activated using the "admin" link next to a user's name in the user's list and search results. The "delete" link was left separate from the admin popup for both ease of use and in order to pass over any loopholes that would needed to be covered after deletion from the popup as well as any conflicts that might arise from the same problem.

Upon clicking the admin link, a modal is activated that is given basic information on the selected user to be used to search the database for related information on the user. The admin switches are then shown in the modal body, updated according to their current true/false status, and displayed as a series of checkboxes. The administrator can then check and uncheck the boxes as necessary, press the submit button, and then is returned to the current page. This can be done as many times as wanted without having to update the page, which is extremely useful when needing to update admin permissions for multiple users.



Edit Privileges for User: user1

☐ Admin

☒ Contest creator

☐ Banned

A "Modal"?

This project used Twitter's Bootstrap Javascript library, and the Bootstrap modal type was used to create the popup. Essentialy, a modal is a section of html code that is hidden when a pages is loaded, but can be activated by clicking a link or a button that will cause it to show on the page. In this case, the admin link was used to activate the admin modal, which grays out the screen and shows the section not earlier shown. The modal structure provided by Bootstrap is very easy to use, and has any needed Ajax or Javascript animation built in, so the use of the modal is very simple to code and use.

Difficulties

There were a number of difficulties encountered when creating an effective and usable admin modal. The first was that the admin user was originally restricted to only be able to delete users. Because admin users are the same as regular users except with the admin flag switched to "true" and users cannot modify information of other users, some sleight of hand was needed under the hood in order to get a user with admin privileges be able to edit the current three admin options. Room needed to be made for more admin switches be added at a later time, and yet easily be integrated with the current modal solution.

The real difficulty came when running tests. Ruby on Rails was created with test-driven-development in mind, but the tests needed for the admin modal proved very difficult and were not able to be completed in the time allotted for this jterm. Because the modal relies heavily on javascript, tests were needed to check for visibility and activation. This was beyond our RSPEC testing suite, so the Selenium Webdriver was selected to test the modal. Selenium is able to open a web browser and visually test items on a web page, which is both very helpful and incredibly fun to watch. Unfortunately, this created a number of problems with other tests that were already implemented in other areas of the project. It also had a issues with our SQLite database that made testing impossible when using the web driver. It is a great tool, but a few things need ironing out before it can be effectively used in this project.

To Do

The functionality for user administration is complete. However, the testing is far from finished. Work needs to be done to correctly integrate the Selenium Webdriver into the project. A number of fixes were tried, but so far none have fully succeeded, and some have not worked at all.