OBJECTIVE

Design and implement a performant and elegant backend API for managing user permissions database.

SCOPE

The scope of this task is to build JSON-based, database-backed, API for managing a list of users and associated permissions. There are lots of details in this task description, be sure to follow these instructions accurately. You will “deliver” your application by checking new code and associated documentation into Github. Treat this exercise as if it were a small slice of a part of an actual consulting engagement, you are building this web-stack to serve as the “permissions server” for a large SOA (services oriented architecture) software application which is to be maintained over time by multiple developers. The goal is to quickly prototype an end-to-end app with the basic functionality specified. We will be looking at the generality, simplicity, extensibility, and cleanliness of the code you deliver, as well as the speed with which you deliver it. But this is not a race, develop code as you would if part of our team, balance design, cleanliness, velocity, and refactoring. During this exercise you should pragmatically trade off your time between the normally expected activities including: Documenting your APIs, Providing test coverage, Refactoring code for high-throughput, fast response-time performance, Refactoring code for elegance, clarity, simplicity, etc.

You code should include a careful Read-Me that includes all steps needed to install and start your backend service on a typical UNIX system.

Please perform very frequent commits as you code, and do this first half of the assignment in one continuous chunk of time, so we can review your approach. Feel free to refactor, etc. as you go, we hope to give you lots of room to innovate the best design/coding approach, and let you do your stuff!

DESCRIPTION

Design and implement a set of JSON end-points which allow other parts of a larger application to update and test user access (permissions) over a set of named objects.

The API deals with:
- a set of Named OBJECTS whose access permissions are being controlled,
- a set of USERNAMES which might have different types of permissions.

- a set of GROUPS that those users may be part of,
- a set of PERMISSIONS which specific users or groups of users might have over specific objects.

Objects, Usernames, Groups, and Permissions should all be referred to in the APIs using unique case INSENSITIVE strings. There are no explicit API calls to create or delete these entities, they are implicitly created at the moment when they are first referred to in any API call. The Permissions API should have JSON calls:
- for adding a user to a group
- for clearing all users from a group
- for adding a permission to a user or group.
- for clearing all permissions directly associated with a user or group.
(this clear operation does not affect permissions that are inherited by a user from group membership.)
- for testing if a particular user has a particular permission over a particular object.
- for querying what permissions a particular user has over a particular object.

HERE IS THE RULE ABOUT ASSIGNING PERMISSIONS
A user has a particular permission over a particular object if they have been directly assigned that permission to that object, or if they are part of a group which has been assigned that permission.

EXAMPLE USAGE
Here is a sequence of API calls when illuminate how the API should work:
- Add users called ‘Bob’ and ‘alice’ to the group called ‘administrators’.
- Assign the user called ‘dan’ and group ‘administrators’ the permission called “VIEW” over the “message of the day” object,
- Assign the “modify” permission to administrators group for the “message of the day” object.
- Query for the permissions that Alice has over the “message of the day” object.
(This should return “view” and “modify”.)
- Test if user Dan has the modify permission on “message of the day” object
(This should return False.)

TASK ONE — Design, document, and implement the API above.
- Focus on completeness of the API.
- Provide concise text documentation of the API, but be sure it is complete.
- Provide a script that can be invoked from the UNIX command line to test your solution.
(We will clone your Github repo locally and your permissions stack and test script should “just run” after following your setup instructions)

Here is the second half of our programming assignment. In your remaining hours you should focus on the front-end programming task, and completing any missing portions of the first half of the assignment. Only if you complete these three tasks should you begin on the extra credit task.

As before, all results should be checked into Github, and all should be executable by following your ReadMe.

TASK TWO — Create a Javascript "Permissions Control Panel" front-end webpage for controlling User Permissions.
- This should be a static HTML/Javascript SPA (single page app).
- Entire app (backend/frontend) should be launch able by invoking a shell script from the command line.
- This task should use javascript to connect to the backend you wrote using your JSON API.
- The goal is to produce an Interface with the specified functionality, no effort needs to be expended on styling this interface.
- The control interface page has two scrollable lists, and 4 control buttons:
* The first scrollable list is a list of GROUP-SPEC-LINES. Each Group-Spec-Line begins with a group name followed by a comma separated list of users that are currently assigned with that list.
* The second scrollable list is a list of all USERS and GROUPS along with *ALL* permissions associated with that user/group. Again each line begins with a name followed by a comma separated list of permissions.

* The button titled "add user to group" will prompt the user for a user & group when clicked and update the group as indicated.
* The button titled "add permission to user or group" will prompt the user for a user/group name and a permission to add.
* The button titled "clear group" will remove all users from the group currently selected in the first scroll list.
* The button titled "clear permissions" will remove all *directly-associated* permission from a group or user.
(just to be clear, after this button is pressed, the display will still show permissions indirectly associated

with a user, if they are still in groups with permissions associated.