Minh Nguyen

Final Project CS 496

Teacher - Class application

Demo: http://web.engr.oregonstate.edu/~nguyminh/ Click on file "DemoFinal496.avi" to download

Overview

My simple app is a demonstration for teachers to create a user account and add classes that they teach into their account. User's do not have administration privileges therefore they cannot add what classes there are onto the database, but rather add what classes exist into their list, as well as deleting the class. My backend further has the classes connected to students to show which students are enrolled in what classes, but this was not implemented on the front-end. So the demonstration will only show users logging in and adding classes to their accounts. Due to some difficulties with my inexperience with the mobile platform, I was unable to programmatically create the student's list for the deliverables. But looking at the backend of my code, you can see that students are their own entity and are connected to Classes, which in turn are connected to Users.

My API allows for adding students to classes, and classes to User's. User's will have an array called class_teach which holds the different c lasses that they teach. They will not be able to successfully add classes to their accounts if the class is not yet already created. Likewise, classes have an array of students that are enrolled, but cannot add students that do not exist yet. In order for this to work, students must exist first, then classes, then users. Although it is completely fine to have an empty array of students in classes, as well as classes in Users. User's will be allowed to delete classes from their accounts, and if I had more time, I would have implemented deleting and inserting students into classes.

My demo shows the creation of 2 separate accounts John and Bill and how their accounts differ with different classes they teach. Below is a list of URI's and how they are used in my app.

GET list of users:	http://final496minh.appspot.com/user
GET specific user:	http://final496minh.appspot.com/user/<username>
POST new user:	http://final496minh.appspot.com/user
DELETE a user:	http://final496minh.appspot.com/user <username>

GET list of classes: http://final496minh.appspot.com/class

GET specific class: http://final496minh.appspot.com/class/<classname>

POST new class: http://final496minh.appspot.com/class

DELETE a class: http://final496minh.appspot.com/class/<classname>

GET list of students: http://final496minh.appspot.com/student

GET specific student: http://final496minh.appspot.com/student/<studentname>

POST new student: http://final496minh.appspot.com/student

DELETE a studentr: http://final496minh.appspot.com/student/ <studentname>

PUT student into class: http://final496minh.appspot.com/class/<classname>/<studentname>

PUT class into USER: http://final496minh.appspot.com/user/<username>/<classname>

DELETE class from user: http://final496minh.appspot.com/user/<username>/<classname>

User accounts

Originally, I had planned on having 2 entities, classes and students. And students will be added to classes. To implement user accounts, I created a 3rd entity, User, which has a username and password and holds an array of classes they teach. When I create the name and password using Post methods, I implement the front end by checking what the user enters with a GET command to see if the name and password matches what is in the data store. Only the correct combination of username and password will allow the login button to take you to the next page. Since user accounts have a class_teach array in their entity, adding classes to each user account is done by passing the username to the next page of my app, and using this username variable and build the string to the URI when I make the PUT call to add classes to this username. This ensures that only that username will receive that class.