Gameful Affinity Space Midterm Progress Report Spring 2018

Harsimran Bath (COMS 3998 W sec 028) | Yuval Schaal (COMS 6901 E sec 028)

1. Abstract

In this report we will describe the progress made in the gameful affinity space, class administration user story. The features we have completed so far under class administration are the Class Administration Design (#346), and the Class Creation (#298). In addition, we made the important decision to use Material-UI as the design library and design paradigm for the project. Not only does this provide a consistent and proper design philosophy and user experience on the website, but it also helps in onboarding developers to this framework, which is created by Google, easy to use, and has tons of useful features out of the box.

2. Architecture

We are following best practices in implementing our front-end and back-end. In our front-end, we are continuing to implement AngularJS's MVC design paradigm to build our app. On the back-end, we are using REST API endpoints and GET/POST methods for getting and creating data. Most of this is inline with current implementations as well for consistency.

3. Design and Implementation

For this user story, the instructor wanted to have a nice UI/UX that makes it easy to interact with SAGE and the Class Administration System. To accomplish this, we implemented wireframes created using the Angular JS Material framework. This framework is both a UI Component and a reference implementation of Google's Material Design Specification. We decided to use it because it provides a set of reusable, well-tested, and accessible UI components based on Material Design. References we used often were from thier official website: https://material.angularjs.org/latest/getting-started

3.1 Class Creation

The class creation user story focused on the instructor's ability to be able to create a class so that they can easily associate students with Missions and monitor students' progress. To accomplish this, we first created a class model which is specified in the image below. On the users' sidebar we added classes. By clicking classes, and then going to the top right drop down, instructors can choose to add a class. In the class addition window, instructors can specify the

class's name and description. Once a class has been created, instructors can then add missions from existing missions to the class.
Class Model: name, description, [roster], [missions], and instructorId.

3.1.1 Endpoints

The following endpoints have been created:

1. Add class: `[POST] /instructors/class/:iid`
2. Get all class: `[GET] /instructors/:iid/classes`
3. Get specific class: `[GET] /instructors/:iid/classes/:cid`

4. Training / Collaborating

This semester we have received many requests of fellow students needing help with working on SAGE. Whether it was Material JS or Angular issues, we met up with students often and invited them to see how we work together in the team to accomplish tasks. This led us to believe that students starting to work at SAGE really need a good amount of resources to be able to start off achieving tasks.

5. Limitations & Assumptions

There are still some questions on anonymizing roster and real roster. We are looking into encryption techniques to address this. Another limitation we are looking into is adding students that are not students / users in the system. For this, we are investigating an approach taken by online coding platform Codio, where email is sent to students with a code to join a classroom. Before joining, the user will be required to login or register an account.

6. Future Work

For the rest of the semester, we plan to complete the class edit (#299), class copy (#300), and student roster upload (#302) user stories / features. In addition, we hope to do some quality testing to ensure implementations are stable.