

Class Administration

Final Report Spring 2018

Harsimran Bath (COMS 3998 W sec 028)

Yuval Schaal (COMS 6901 E sec 028)

Table of Contents

Abstract	3
1. Introduction	3
2. Related Work	4
2.1 Khan Academy [3]	4
2.1.1 Adding a Class	4
2.1.2 List of Classes	5
2.1.3 View and Edit a Class	5
2.2 Edmodo [4]	6
2.2.1 Create a class	6
2.2.1 Edit a Class	7
2.3 Material-UI for AngularJS [5]	7
3. Accomplishments	8
3.1 Material UI Design Paradigm	8
3.1 Class Creation	9
3.1.1 Create a New Class	10
3.1.2 Copy an Existing Class	10
3.2 Class Selection	11
3.3 Class Edit	12
3.4 Adding Missions to Class	12
3.5 Mission Management	13
3.6 Add Roster	14
3.7 Manage Roster	15
3.8 Upload Roster	16
3.9 Email New Students	17
3.10 Delete a Class	19
4. Documentation	19
5. Future Work	19
5.1 Material-UI on SAGE	19
5.2 Front-End Code Quality Improvements	20
6. Conclusion	22
References	23

Abstract

In this paper, we report our accomplishments in implementing the Class Administration feature in SAGE (Socially Addictive Gameful Engineering)[1]. First, we discuss related works to give a better understanding of the implementations of class administration feature by similar projects. Then, we describe the scope of the class administration feature in the context of SAGE. Next, this paper describes the tasks implemented: class creation, class editing, copy class, class deletion, and upload student roster features that have been completed throughout this semester. Finally, we will describe potential future work. The contents described in this report are part of the Gameful Affinity Space epic in TFS and the Class Administration feature.

1. Introduction

The motivation behind the SAGE Gameful Affinity Space is to create a platform with a user friendly and engaging interface that allows instructors to easily create different learning curricula in computational thinking for their students. According to Jeannette Wing, “computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science”[2]. This takes researcher’s work in the Gameful Affinity Space and adds a higher degree of gamification for the students.

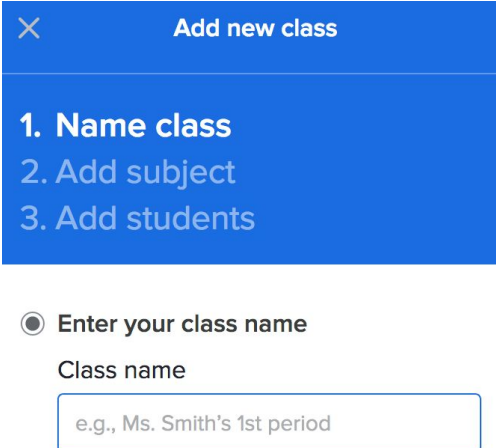
To further enhance the platform, we have implemented the creation and editing of classes in the instructor dashboard. This feature allows teachers to host classes in which students can be enrolled in to learn gameful thinking techniques. Furthermore, teachers can assign missions to a class to create a designated curriculum for that specific class. Instructors can now create and manage their classrooms.

2. Related Work

2.1 Khan Academy [3]

Khan Academy is an existing online learning management system that provided inspiration for our work. It has a dashboard for teachers to manage classes, similar to what we built. The purpose of classes in Khan Academy is to enroll students in a specific class room and build a curriculum for the classroom using existing subjects in Khan Academy. Our project works similarly to this.

2.1.1 Adding a Class




The image shows a blue modal dialog titled "Add new class" with a close button (X) in the top left corner. Inside the dialog, there is a list of steps: "1. Name class", "2. Add subject", and "3. Add students". Below the dialog, there is a radio button selected next to the text "Enter your class name". Underneath this, the label "Class name" is followed by a text input field containing the placeholder text "e.g., Ms. Smith's 1st period".


In this user interface (UI), Khan Academy allows the instructor to create a classroom for students. In this UI, the instructor specifies the name, the coursework, and students for the class. The students are then associated with this specific class, run by this instructor, and progress through the curriculum.

2.1.2 List of Classes

Your classes



Mathematics 101: Multiple subjects
0 students



Science 101: Multiple subjects
0 students

In this screen, Khan Academy has a dashboard for viewing and managing existing classes. It also gives the instructor a statistic on the number of students in the class. SAGE Class Administration could use similar structure to display more information about the class. Right now the instructor is required to visit the class.

2.1.3 View and Edit a Class

[Assignments](#) [Progress](#) [Activity](#) [Roster](#) [Settings](#)

Assignments for Mathematics 101: Multiple subjects

[Active](#) [Past](#) [Saved](#)

ASSIGNMENT NAME	DUE DATE & TIME ▼	ASSIGNED ON	COMPLETED
You have no active assignments	—	—	—

[Find Early math content to assign](#)

Khan Academy has a dashboard for viewing and editing a class. On the **Assignments** tab, the instructor can manage the assignments in the curriculum of this course. This including,

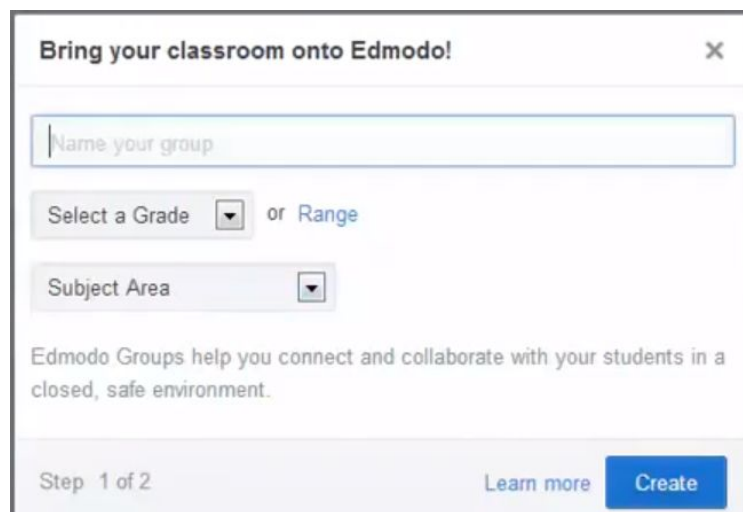
adding and removing assignments. Furthermore, the **Progress** tab allows the instructor to visualize the progress of students in the class through charts. The **Activity** tab allows the instructor to view statistics of student engagement in the classroom, such as amount of time spent on assignments. The **Roster** tab is where the instructor can add or remove students from the classroom. Finally, the **Settings** tab is where the instructor can manage the properties of the classroom, such as name. SAGE Class Administration shows tabs similar to Assignments, Progress, and Roster, while settings are available in Edit Class.

2.2 Edmodo [4]

Edmodo is one of the world's largest K-12 social learning community where teachers, students, and parents can connect safely and securely. Edmodo has groups, which are similar to classes in SAGE. Therefore, Edmodo also served as a source of inspiration for our implementations.

2.2.1 Create a class

The figures below show the group/class creation process in Edmodo.



The screenshot shows a web form titled "Bring your classroom onto Edmodo!" with a close button (X) in the top right corner. The form contains the following elements:

- A text input field labeled "Name your group".
- A dropdown menu labeled "Select a Grade" followed by the text "or Range".
- A dropdown menu labeled "Subject Area".
- A paragraph of text: "Edmodo Groups help you connect and collaborate with your students in a closed, safe environment."
- A footer section with "Step 1 of 2" on the left, a "Learn more" link in the center, and a blue "Create" button on the right.

You're almost there! ✕

Expected Group Size What Is It?

Describe your group - Max. 260 characters

Answer these last few items to help us create the best Edmodo Group experience for you and your students.

Step 2 of 2 [Learn more](#) [Finish](#)

2.2.1 Edit a Class

Group Settings ✕

Test Group

Higher Education or Range

All


25+ What Is It?

☐ Default all new members to read-only

☐ Moderate all Posts and Replies

This is the group that I am just showing for tutorials...

[Archive Group](#) [Delete Group](#)

 [Save Settings](#)

2.3 Material-UI for AngularJS [5]

A major focus for us initially was to find a user interface philosophy to bring consistency to the platform, as well as have a well-maintained UI library to make it easier to implement

features. Material-UI is a library that fits our goals and is supported by Google. Therefore, this is now a part of the SAGE project. This project is also very well documented, so it will make it easier for future students to build UIs using this library.

3. Accomplishments

This semester we focused on building the class feature that allows instructors to organize students within a class. This enables instructors to easily assign missions to students and potentially track their progress. We implemented all the UI elements to make this possible, as well as the routes, back-end APIs, and MatLab data models to make this possible. Overall, we succeeded in our project and closed all our assigned user stories.

3.1 Material UI Design Paradigm

As evident in figure 5 below, we have successfully added Material-UI library as a dependency into the project. As shown, this has the potential to provide consistency to the UI and a lot of pre-made components to aid future researchers in designing the rest of the SAGE platform, to be used by teachers and students in the community for testing in the very near future. The documentation will help in building UIs and the built-in components will remove complexity of building UIs from scratch.

Programming Loops

EDIT CLASSDELETE CLASS

ROSTERMISSIONSMETRICS

DELETEADDIMPORT

Roster

<input type="checkbox"/>	Student Name	Email
<input type="checkbox"/>	Bob	bob@test.com
<input type="checkbox"/>	Mary	mary@test.com
<input type="checkbox"/>	Sam	sam@student.com

Rows per page: 51-3 of 3

Figure 1: Classes View

3.1 Class Creation

A major motivation for this project is the ability to build classrooms in SAGE so that the instructor can add students and manage the curricula of the class. This is relatively easy to do. Essentially, the instructor can go to the dropdown on the top and click 'Add Class.' This brings up a popup asking the instructor to create a New class or Copy a class (Figure 2). The instructor can choose to create a new class from scratch, or create a new class based on an existing class.

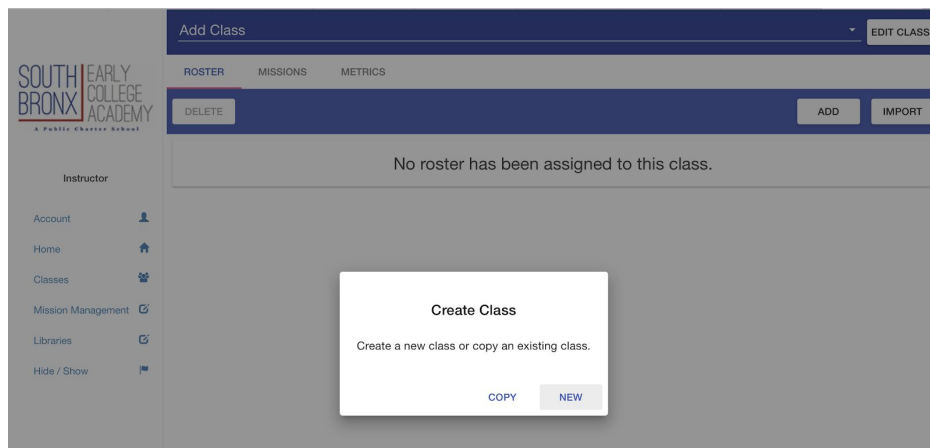


Figure 2: Create Class View

3.1.1 Create a New Class

When the instructor clicks 'New', they are presented with a modal to enter the Name and Description of the classroom they want to create (Figure 3). This form also enforces validation rules, to ensure Name is required and Description stays within 150 characters (this can be changed). The user can cancel out of this modal or click 'Finish' to create this classroom. Once created, the page will automatically update and select this newly created classroom.

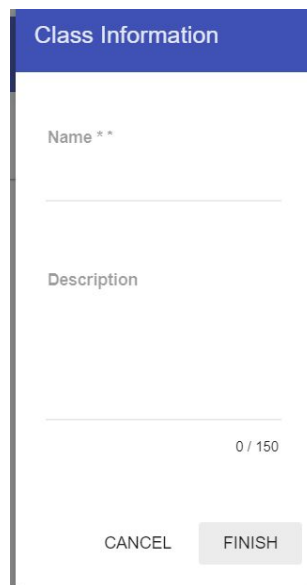


Figure 3: New Class Creation View

3.1.2 Copy an Existing Class

To make creating classes easy for instructors, we have added a feature to be able to add a class by copying another, already previously created, class. As seen in figure 3.1, the class copy option allows the instructor to create a class by choosing from the dropdown menu from a list of existing classes. The instructor is able to add their own name and description for their newly created class. An instructor is also able to choose whether they want to copy either

the roster, missions, or both from the existing class by using the checkboxes at the bottom of the class copy view.

Create New Class from an Existing Class

Name **

Mathematics

Description

This class builds on the principles of algebra in Programming.

62 / 150

Another Class

☒ Copy All Students

☒ Copy All Missions

CANCEL FINISH

Figure 3.1: Class Copy View

3.2 Class Selection

As the instructor creates their classes, it becomes important for them to easily find and navigate between those classes. While clicking 'Add Class' brings up the popup to create a new class, the instructor can select an existing class in the dropdown, and it will immediately update the UI with that class's information. AngularJs' Single-Page-Application model allows us to build this dynamic and efficient experience, without requiring server postbacks. Switching between classes is, hence, very fast.



Figure 4: Class Selection View

3.3 Class Edit

To edit the previously entered data of a class, the user can click the 'Edit Class' button on the top right. This will bring up a popup where the user can change the name and the description of the class (Figure 5). This form, similar to the create class form, also enforces validation rules to ensure the integrity of the data. Finally, once the user clicks 'Finish', the updated name immediately reflects in the UI, confirming the change.

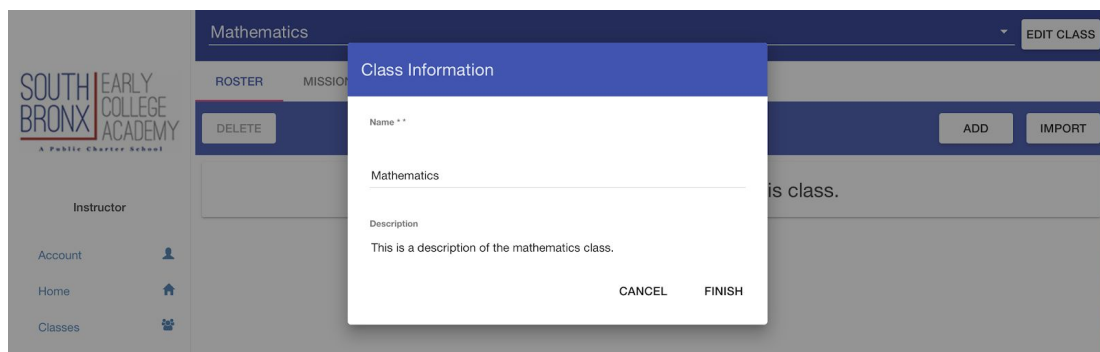
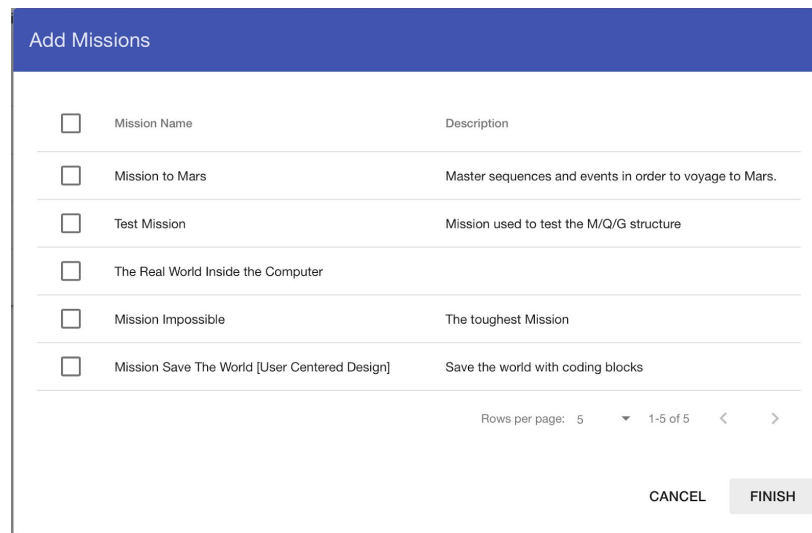


Figure 5: Class Edit View

3.4 Adding Missions to Class

We added the feature for instructors to be able to add missions to a class. As shown in figure 6, an instructor can view all the missions available that can be added to a specific class. A selection of one, multiple, or all missions can be done in this view and the action of adding the selected missions to the class can be done. A total of five missions are displayed at a time in this view (this can be changed via the toolbar at the bottom). The toolbar at the bottom has

arrows which allow the user to switch between pages. Once a mission has been added to the class, that mission will no longer appear in the 'Add Missions' view.



The screenshot shows a modal titled "Add Missions" with a blue header. Inside, there is a table with two columns: "Mission Name" and "Description". Each row has a checkbox in the first column. The table contains five rows of mission data. Below the table, there is a pagination control showing "Rows per page: 5" and "1-5 of 5" with navigation arrows. At the bottom right, there are two buttons: "CANCEL" and "FINISH".

<input type="checkbox"/>	Mission Name	Description
<input type="checkbox"/>	Mission to Mars	Master sequences and events in order to voyage to Mars.
<input type="checkbox"/>	Test Mission	Mission used to test the M/Q/G structure
<input type="checkbox"/>	The Real World Inside the Computer	
<input type="checkbox"/>	Mission Impossible	The toughest Mission
<input type="checkbox"/>	Mission Save The World [User Centered Design]	Save the world with coding blocks

Rows per page: 5 1-5 of 5 < >

CANCEL FINISH

Figure 6: Adding Missions to Class View

3.5 Mission Management

After adding missions to a class, the instructor has the power to remove missions from the class. This functionality is useful for the instructor when they want to change the curricula of their class. They can select one or more missions in the table and click the 'Delete' button to remove the selected missions from the class (Figure 7). These deleted missions will now appear again in the 'Add Missions' modal. Furthermore, the table has pagination capability in case there are many missions. This makes it easier for the instructor to navigate through the missions. The page size is also configurable.

ROSTER MISSIONS METRICS		
DELETE		ADD
Missions		
<input type="checkbox"/>	Mission Name	Description
<input checked="" type="checkbox"/>	The Real World Inside the Computer	
<input checked="" type="checkbox"/>	Mission Impossible	The toughest Mission
<input type="checkbox"/>	Computational Thinking	Series of courses to develop CT concepts
<input type="checkbox"/>	Parallel Thinking	CT concepts
Rows per page: 5 1-5 of 4 < >		

Figure 7: Mission Management View

3.6 Add Roster

Another important part of creating classrooms is adding students to the class. We have built a very simple and advanced UI to ensure that it is easy for teachers to add students to a class, especially since there can be thousands of students in the system. When the user clicks 'Add' in the roster tab, the 'Add Roster' modal appears on the screen (Figure 8). In this UI, the user can search for a specific student among all students in the system. The user can also use pagination to manually find students. Finally, the table supports multi-selection, so the instructor can add multiple students to the class. All these features will help the instructor easily add students to their classes. Once students are added to a class, they will no longer appear here until their are manually removed from the class.

<input type="checkbox"/>	Student Name	Email
<input type="checkbox"/>	Sam	sam@student.com
<input type="checkbox"/>	Dean	dean@test.com
<input type="checkbox"/>	Mary	mary@test.com
<input type="checkbox"/>	John	john@test.com
<input type="checkbox"/>	Bob	bob@test.com

Rows per page: 5 1-5 of 41

CANCEL FINISH

Figure 8: Add Roster View

3.7 Manage Roster

At the top of the classes section, there is the 'Roster' tab. When the instructor clicks on this tab, they will be shown the roster management view as shown in figure 9. In this view an instructor can view and select students that have been added to the roster. Once the students are selected in this view, an action of delete at the top left can be taken to remove the selected students from the class roster. At the top right of the roster management view, there are add and import buttons that have the actions discussed in sections 3.6 and 3.8, respectively.

ROSTER		
MISSIONS		
METRICS		
<div>DELETE</div> <div>ADDIMPORT</div>		
Roster		
<input type="checkbox"/>	Student Name	Email
<input checked="" type="checkbox"/>	Bob	bob@test.com
<input checked="" type="checkbox"/>	Mary	mary@test.com
<input type="checkbox"/>	Sam	sam@student.com
<div>Rows per page: 5 1-3 of 3</div>		

Figure 9: Manage Roster View

3.8 Upload Roster

Another important use case is that often times it might be hard to add students to the class, simply because there are too many students in the system. To circumvent this, we have introduced the option to Upload a CSV file with all the roster the instructor wants to add to the class. This adds a very simple and efficient way to add students to a class in SAGE. The format of the file is described in the text in the UI (Figure 10). The instructor can upload a three column CSV file with students' first names, last names, and emails. Once the file is dragged inside the box, the instructor can click 'Finish.' At this point the file is uploaded and processed on the backend. Students whose email addresses already exist in the system will be automatically added to the classroom. Students whose email addresses do not yet exist in the classroom will be invited to the class via an email.

Upload Roster from Csv File

File Added: roster.csv

Please upload a Csv file containing students to enroll. The first three columns of the CSV file must be: First Name, Last Name, Email, The first row of the Csv file will be treated as the header row and will be ignored.

Drag your Csv file here

CANCEL FINISH

Figure 10: Upload Roster View

3.9 Email New Students

After importing the CSV, the SAGE system generates an “eviteToken” for each student which does not exist in the system. Then, an email is sent to those students via their email addresses, inviting them to join the class. An example of an email sent to such user is shown in figure 11. The email mentions that the student has been invited to join SAGE and to register for an account. In the backend system, the user token sent by email, as well as the student's email, and the class id get put into the invitePending table, and the row in the table gets removed once that user is registered.

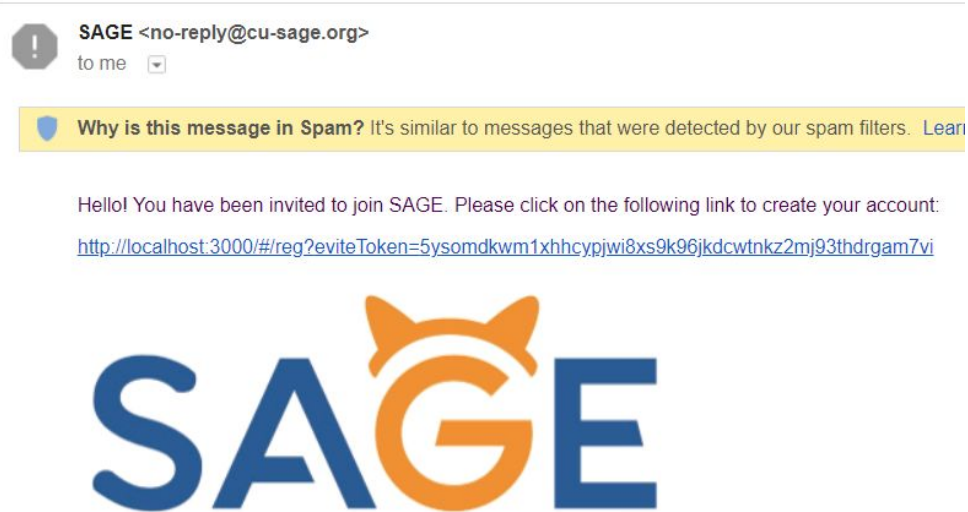


Figure 11: Received Email

As shown in figure 12, the link in the email takes the student to a registration page that automatically chooses their role as a student and includes an eviteToken in the url. After the user has registered, they will be put into the corresponding class's roster and the user will no longer be in the invitePending table.

The image shows a web browser window with the address bar displaying 'localhost:3000/#/reg?eviteToken=s5hiibubk3lmezdc7ocac3dik...'. The browser tabs show 'gmail', 'COMSE 6998_014_20', 'columbia university', 'Robotics Book', and 'Tech Cor'. The main content area shows a registration form titled 'Sign up' with a user icon. The form has three input fields: an email field containing 'newstudent@gmail.com', a password field with a yellow background and a 'Please enter your email' placeholder, and a username field. Below the fields is a blue 'Sign up' button. At the bottom of the form, it says 'Already have an account? Sign in now'.

Figure 12: Registration of New Student by Email, Showing eviteToken

3.10 Delete a Class

Another important feature we have added to the class portion of SAGE is the ability for an instructor to delete a specific class. As shown in figure 13, an instructor can choose a class and then use the 'Delete Class' button on the far right to mark the class as deleted. In the backend, the class is never actually deleted from the database. Instead, a flag of `isDeleted` is simply set to true. Thus, the class is no longer shown to the instructor. Though, any researcher could log into the database and see that the class still exists. This opens the possibility of “un-deleting” a classroom in the future, in case the instructor desires to do so.



Figure 13: Delete Class Button on the Right

4. Documentation

Documentation for all endpoints created for classes can be found here:

<https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/390922243/Class+Administration+Documentation>

5. Future Work

5.1 Material-UI on SAGE

Since we have introduced Material-UI to SAGE, potential future work is to convert rest of the system from the existing Bootstrap implementation to Material-UI implementation. Although this is added work, it has many long term benefits. First, the components in the Material library

will remove some of the complexities in current UI that are manually created. Secondly, future students will tremendously benefit from an existing documentation and pre-built components that are thoroughly tested by Google. Future students can focus on building their projects instead of working out different UI implementations. Thirdly, Material brings out conformity and consistency to the look and feel. Therefore, future work is to change all existing UI to use Material-UI and completely remove Bootstrap's dependency.

5.2 Front-End Code Quality Improvements

The implementation code in the front-end is at times hard to understand and follow, and there are not any strict conventions being followed. We can do the following things to improve this, and begin working towards professional code quality, which can be enforced through DevOps:

1. EditorConfig[6]: This is a very useful tool that enforces coding style, such as indents, braces, new lines, and so on. Most code editors already support this, and this will help bring consistency to the coding style. The editor automatically utilizes these, so the code always follows the style laid out in the editor config file.
2. ESLint[7]: This is another powerful tool that ensures the quality of the code and enforces good coding conventions. It can also at times catch possible errors, or suggest better coding suggestions. There are preconfigured ESLint guides, like Airbnb's and Microsoft's, that will enforce their coding conventions and styles.
3. DevOps: ESLint can be configured into the Daily/CI build, as well as, directly into the Git pre-commit. While the online builds will fail if ESLint fails, the Git pre-commit will prevent students from committing code with ESLint errors.

4. Angular Application Architecture: Right now, the controllers have no consistency in how they are written. Many controllers are directly making Http calls. We can improve this design by breaking off Http calls into separate services that can be injected into the controllers. This can enable us to introduce caching into the system, as well enforce single responsibility so that we can better unit test our code.
5. Back-end Application: The express node application right now has commented code and routes in different files. The routes also do not have a well-defined URL structure. We can introduce some structure to the back-end by splitting into different files according to routes. The URL routes can be updated to be more semantic. Lastly, the routes are directly calling the Data models, it would be better to introduce some sort of proxy functions or adapter to encapsulate the data from the APIs. This would allow us to unit test the APIs. For example, this is a good REST API semantic. For managing a class:
 - a. [GET] /instructors/:iid/classes/ - This lists all classes
 - b. [POST] /instructors/:iid/classes/ - This creates a class
 - c. [PUT] /instructors/:iid/classes/:cid/ - This edits a class.
 - d. [DELETE] /instructors/:iid/classes/:cid/ - This deletes a class.

This API model is easy to understand and no matter what endpoint there is, we always know what to expect.

6. Build Pipeline: Lastly, it would be nice to use a build pipeline like Webpack or Gulp to compile the sources. This has many long term benefits, including code compression, modularizing a single page application to dynamically load modules when needed, source mappings for debugging, and optimizations for production builds, like minimization and obfuscation.

These some suggestions that would help the quality of the code and bring in more conformity.

6. Conclusion

The contributions we have made this semester of adding Class Administration to the SAGE platform have been extremely crucial to future instructors of this system. Instructors now have a way to group missions in the form of classes and be able to add students. The instructors can also invite students to the platform and to their classrooms, if the students are not yet registered. Finally, by assigning missions to the class, the instructor can create class-specific curriculums to best teach their students. Throughout the process of working on creating classes, we have integrated Material-UI into the SAGE system. Material-UI allows other researchers to more easily create pages that are visually consistent to the rest of the platform. Overall, we met all our goals and completed all our deliverables for the Class Administration feature. Additionally, we implemented email invites and Material-UI, which were out of scope, but we felt they added substantially to the project's success.

References

[1] Bender, J., "Tooling Scratch: Designing a Collaborative Game-Based Learning System to Infuse Computational Thinking within 6-8 Curricula," Columbia University, New York, NY 2015.

[2] Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3), 33-36.

[3] "Khan Academy." *Khan Academy*, www.khanacademy.org/.

[4] "Edmodo." *Edmodo*, www.edmodo.com/.

[5] <https://material.angularjs.org/latest/>

[6] <http://editorconfig.org/>