Student Portal Manager

A web-based application that manages projects for software engineering/development courses.

Description

Primarily using Node.js, Express.js, and a MySQL database, this web application allows professors/clients to post projects available to specific classes and students in that class can order their preferences among all other available projects. The professor will then be able to assign projects to all students in a class based on a provided algorithm.

Installation/Running the application

In order to run the application, the system must have a MySQL database running (the schema is provided). Node.js and Express.js must also be installed. Open the .env file to enter the information for the running database. At this point, the application is ready to be run. Open the software folder in a terminal window, and run the command: node app.js to start the application.

Code Structure

Brief description of the code each folder contains:

- controllers: These files are responsible for actions that interact with the database.
- public: These are the two .css files that stylize the entire interface.
- routes: These .js files contain the function calls for all actions on the website.
- views: These .hbs files (essentially .html files) are the structure for each page on the website

Function Descriptions

Structure of descriptions:

Folder:

| **File:**
| | (*) Function: Description.

(*): If there are parenthesis before a function name, the character within them denotes whether the function is only available to use when in the student/professor view (S for student, P for professor). If no character is provided, the function can be used in either view.

controllers:

auth.js:

register: Gets information from textboxes on registration page. If the email is not already registered, creates new user in database with provided information.

login : Queries the database for information provided on login form. If the user exists and the provided password is correct, signs the user in with the user's role.

courses.js:

(P) addcourse	: Inserts a new course into the database with provided information.
(S) getEnrolledCourses	: Returns all courses that a student is enrolled in.
setting	: Used to help render the settings page with the username and email fields already filled in.
updateSetting	: Updates user information in database with new information provided by the user on the webpage.
(P) createcourse	: Creates a new course with provided information.
(P) viewcourses	: Displays all courses created by professor.
(P) deletecourses	: Deletes selected course.
(P) getcourses	: Displays all courses.

projects.js:

(P) addproject	: Inserts a new project into the project database with all the information provided by the user on the webpage.
(P) viewprojects	: Queries the project database for all projects and displays them for the professor.
(P) deleteproject	: Deletes selected project from the project database.
viewsingleproject	: Displays all information about selected project. Works for both professor and student log ins.
(P) updateProject	: Updates information in the project database for selected project.
(S) getProjects	: Queries the database for all projects in classes a student is enrolled in.
(S) select_project	: Used for submitting student project preferences.
(S) submitprefs	: Submits a students project preferences to the database.
(S) getStudentProjects	: Displays all projects a student has provided preferences for.
(P) assignProjects	: Gets all student preferences and uses provided algorithm to assign students to projects based on those preferences.

routes:

auth.js:

There are 3 post/get functions in this file that are called by actions on the webpage: register, login, and logout. These functions simply navigate the user to the correct page bases on which input they clicked on.

pages.js:

This file is responsible for rendering the page based on user navigation.

professor.js:

This file is responsible for professor specific navigation and rendering. These are called through events on the webpage.

student.js:

This file is responsible for student specific navigation and rendering. These are called through events on the webpage.