Project 4

Brief Description of the Project:

This project is built on ruby on rails and is used to assign student graders to a list of courses in the CSE Department of Ohio State. The interface of this program first works with a login/signup page where the student graders, administrators and instructors can either sign up or login. Student graders can signup/login to fill out the application to be a grader for the course they are interested in by filling out certain fields. They can also login and choose to edit their application. The administrators can login to see a list of applicants and select a particular applicant to be a grader for a particular course. The instructors can login/sign up to fill recommendations for certain students. For this project the link we are using to scrape data from is "https://web.cse.ohio-state.edu/oportal/schedule_display".

Prerequisites:

- Ruby 2.7.1
- Rails 6.0.3.2
- Gemfiles: **Nokogiri, HTTParty, Devise**
- ActionText

Installation Steps to get the Project Working:

- First git clone our repository using **'git clone URL'.**
- Install gems and set up database using **'bundle install'** and **'db:migrate'**
- **Run 'yarn install --check-files' to make sure that your yarn packages are up to date.**
- **To run the application, type in 'rails server' or 'rails s'**
- Lastly go to **'localhost:3000'** on your browser to use the application

Assumptions Made:

- Students could manually enter when he/she is available.
- Admin could assign graders to sections by adding people to the section grader database.
- Student, Admin, and Instructors will not select one of the other choices for their login

Design:

This project has used Bootstrap for styling and is broken down in the following MVC components:

1) **Model**- Our application has used SQLite to create grader application form, recommendation form, course sections, section grader sets, and the evaluation form**.** A table for the list of courses is created when the scrape button is clicked. A table to store information for instructor recommendations is created. A table to store information for login details is created. A table to store information for grader applications is also created. A table to store how the student did as a grader is also created. Various regular expressions and constraints were described in the models to handle the proper formatting and the ranges.

2) **View-** When the application is run then a login/signup page is loaded where the users can enter their details and credentials to either signup or login as either a student grader, administrator or an instructor.

   For a student grader login, when the student logs in he/she sees a view with a form with certain details like his/her name, dot number, email id, course that they are interested to grader for, GPA, available times to work, GPA and qualifications. He/She has to fill out and submit this form to apply to be a grader for a course. At any moment he/she can also login to edit their application.

   For an administrator login, when the administrator logs in he/she will be able to see the list of CSE courses offered by the CSE Department and also see the list of grader applicants for particular courses. They can choose to assign a grader for the particular courses.

   For an instructor login, when the instructor logs in he/she will see a view with a form that they can fill out to submit a recommendation for a particular student. They too can login at any moment to edit their recommendation. They could do an evaluation of a particular student. The action text tool was used to support paragraph length texts like the recommendation letter.

3) **Controller-** The root route goes to the home page with the student, admin, and the instructor buttons. There are controllers for the evaluation form, the grader application form, the instructor recommendation form, the graders, and the course list. Each of those handles the show, edit, and destroy and create a new instance of that database.