Authors: Grant Zimpfer, Blake Perkins, Siddhant Gangwani, Yi Chen, Gabe Kornick

Web Scraper

1. Basic Functionality

The application takes the entire CSE department course listing, and pulls it into a SQLite database that the rails component utilizes to generate the model for the user. When opening the rails server on the local host, the user is first prompted to sign-up for an account or login to an existing account. Once a login has been created, the user can then see a list of every class with their respective section number, times, instructors, and other details from the course website. The user can then select courses to have a more focused view of the course or edit it. The user can also choose to delete a course with the destroy button, and add a course using the new course section button at the bottom of the page

2. Technical Requirements

Ruby Version: 2.7.1p83
Rails Version: 6.0.3.4
Nokogiri Version: 1.10.10
HTTParty Version: 0.18.1
Devise Version: 4.7.3
SQLite Version: 1.4.2
URL that we scrape the data from:
https://web.cse.ohio-state.edu/oportal/schedule_display/index?utf8=%E2%9C%93&strm=1208&catalog_nbr=&location=&instructor

- Open the courses folder within project_3 and run ‘bundle install’ to install the necessary gems for the application.
- You need to run yarn installation through the node modules using “sudo npm install --global yarn” followed by “sudo yarn install --check-files”
- Run “rake db:setup” and “rails db:migrate” to scrape the data and save it to the local database
- Then, run ‘rails server’ and go to localhost:3000 in a browser to run the application.
- Sign up with an email and password to view the scraped data

3. Implementation

We implemented the web scraper using Ruby with the help of the HTTParty and Nokogiri gems. HTTParty pulls the raw html down and Nokogiri parses it into its components that can be easily retrieved for the database. This can be found in seeds.rb in the database folder. Every course is scraped into its own class with its attributes as components, and is put into an SQLite database, that is migrated to the Rails application where the controller creates course

displays for the web-page. The web page's first page forces the user to create a login which is handled with the Devise gem, as it logs the user's info into a model that authenticates future logins. Once logged in, the user can edit (update), show (display), and destroy (delete) the courses that were scraped or add a new course through the create button at the bottom, which were all implemented in the course sections controller.

4. Architecture/Frameworks

Model: We are using SQLite to store our data. We have 3 models- application_record.rb and course_section.rb (for storing information about the courses), and user.rb (for login/logout)..
View: We displayed the data into a table shown in the webpage index.html.erb file in the course_sections folder. We also displayed the login form in the show.html.erb file as well as the application.html.erb in the layouts folder.
Controller: We have 2 controllers - application_controller.rb and course_sections_controller.rb.