

Rate My Courses

Project Report

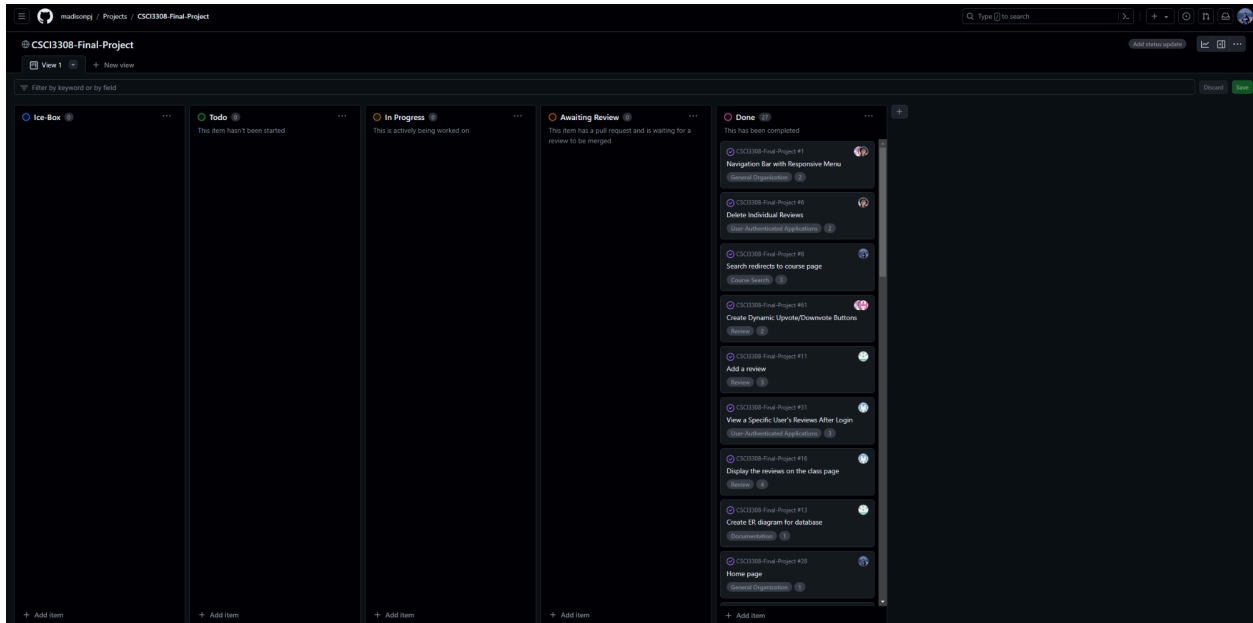
Matt Harper, Dalton Prokosch, Maddie Kloud, Madison Jones, Donghae Yi, Ori Grushka

Description

Rate My Courses is a spinoff of the popular *Rate My Professors* but aims to provide students with a platform for sharing their opinions about classes instead of people—how much homework will I get every week? How is the course structured? What content do we actually learn? How do we learn that content? All of these questions can be answered through the collective feedback of students who have taken the course and *Rate My Courses* provides a place to collect and access that information. FCQs are offered by the school but have no way for students to freely express opinions, warnings, or advice for other students to see. *Rate My Courses* has users rate courses by metrics but also has a place for users to write comments without restriction. Reviews are then self-moderated by other users with a rating system that ensures malicious or hateful comments are quickly hidden from the view of others. In addition, the user can filter the reviews on a course page to easily find the reviews that are most relevant to them. Overall, *Rate My Courses* is a platform for students, by students—it aims to be a resource where students can share their opinions on courses they've taken and find the courses that will resonate with them most at their university.

Project Tracker

<https://github.com/users/madisonpj/projects/1>



Video

<https://youtu.be/cdr9eSnpycM>



VCS

<https://github.com/madisonpj/CSC13308-Final-Project>

Contributions

Matt: @matt-harp

I worked on the class search, which included querying the CU course API and displaying those results in real time. That required figuring out client-side scripting within Handlebars to debounce queries and a CORS workaround to fetch course information from the school. I then created the route and page to view courses, created the functionality for updating reviews, and improved the styling on the account page.

Dalton: @garthable

The primary systems I created were the user authentication system, the voting system, and the sort system. The authentication system was responsible for creating user accounts, logging users in and logging users out. The voting system was designed as a way for users to help curate the content of our site. This required me to create the upvote and downvote buttons that generate votes, and modify SQL queries to display the vote amount. The sort system orders the content of a course page either by most positive or negative review, most upvoted review, or most recent review.

Maddie: @maddiecloud

I worked on the database initialization and table relations using PostgreSQL. I also created a feature that enables users to write a new course review. I implemented this by making a get route in index.js that renders the review page (which I created using handlebars/HTML/CSS) whenever the user clicks the "write review" button on a course page. I made the review page a form post method to populate the reviews table with the user's inputted review data. In addition, I worked on the overall styling of the application using Bootstrap libraries.

Madison: @madisonpj

I focused primarily on the delete review feature. This allows users to delete reviews that they have posted. This included the route in the index.js file and the implementation in the user's account page. Beyond this, I also contributed to navbar. For the navbar, I created all of the links to route to different pages and included a feature so the available links are dependent on whether the user is signed in. I also styled the navbar. Beyond these specific features, I also assisted with the overall styling and consistency of the webpage.

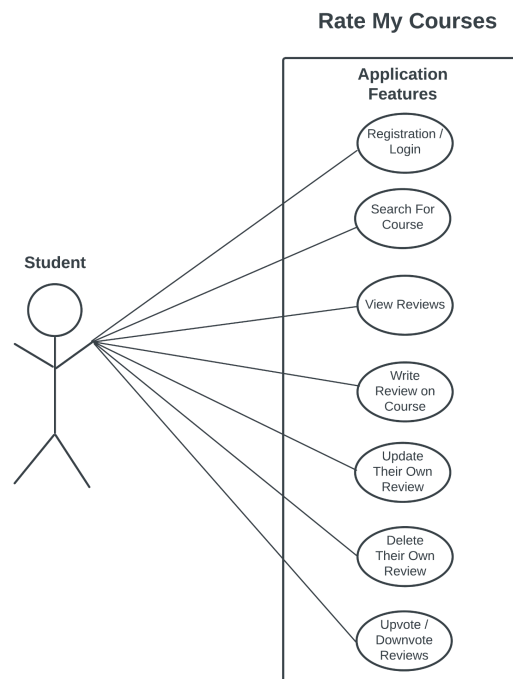
Don: @DonghaeYi

I worked on displaying the account page. This included creating the API route to render the account page as well as making sure I was fetching data from the database to show the user's reviews when they were logged onto the application. The review included the ratings as well as querying the user's username and review itself. I also created dummy data mainly for the presentation, so that we can demonstrate what the account page and course pages look like. Beyond this, I helped out with more general things such as slides/notes, debugging, and other.

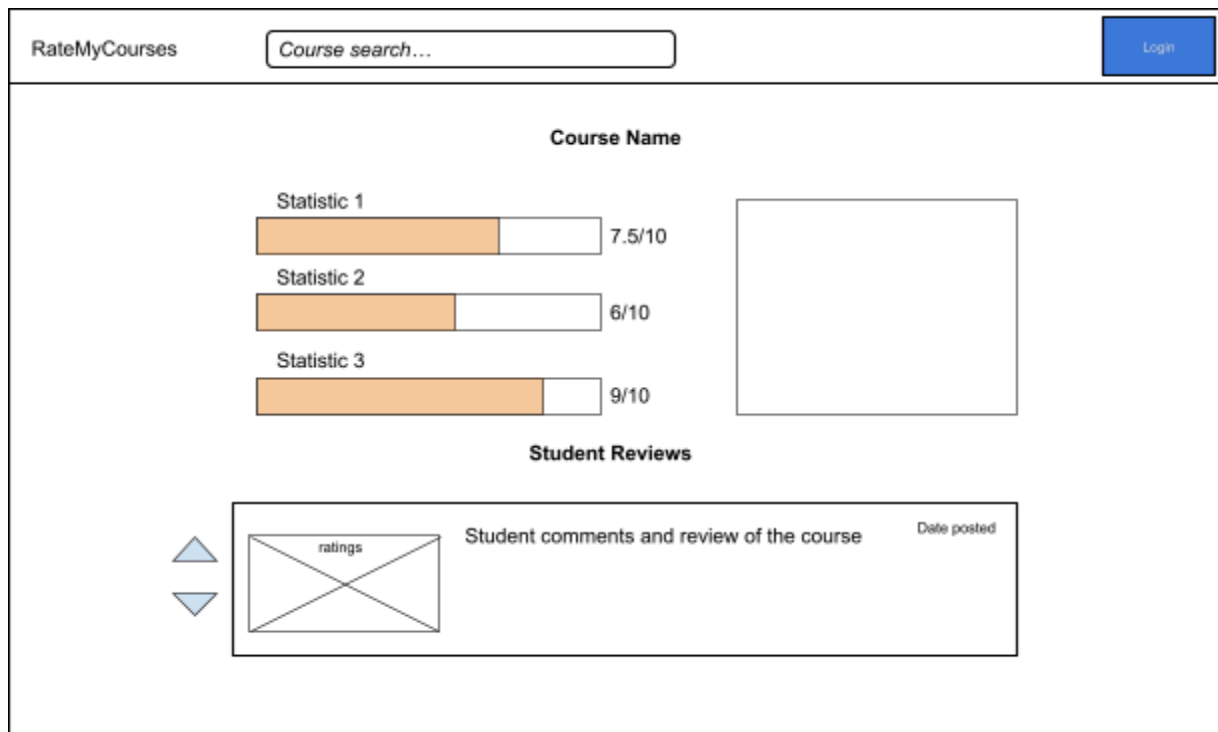
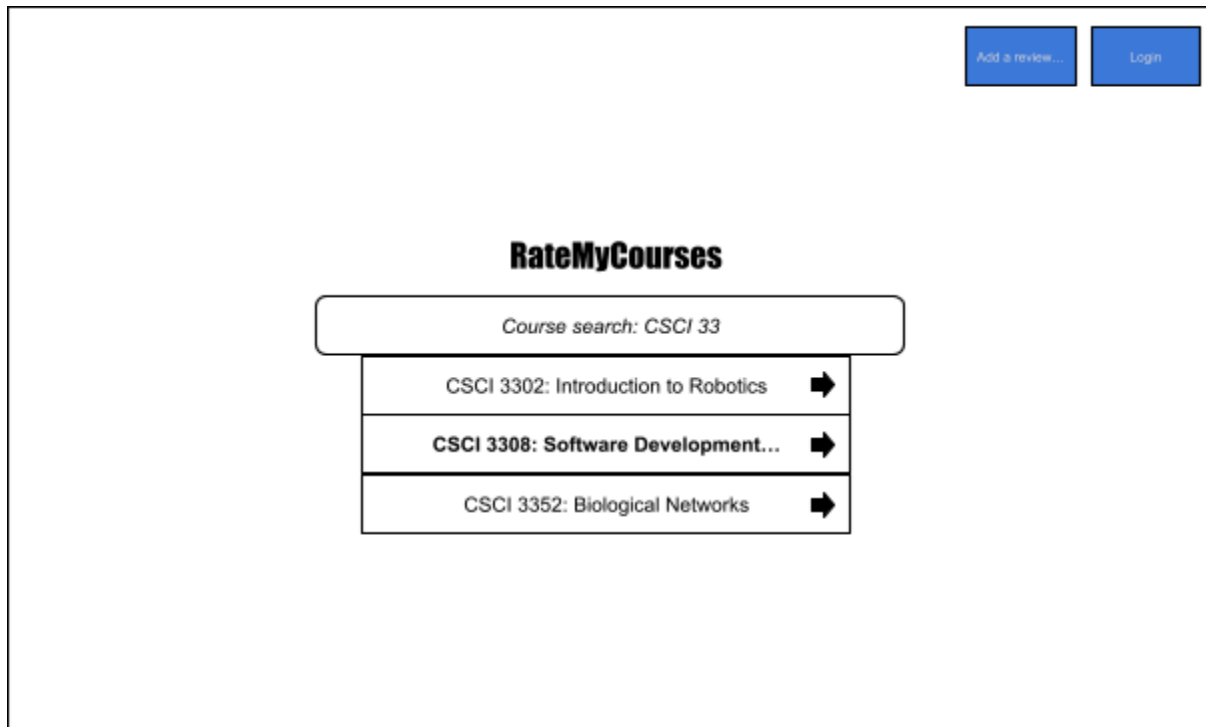
Ori: @origrushka

I primarily worked on the voting system alongside Dalton, figuring out the functionality behind selecting and deselecting the buttons to ensure that votes are removed upon deselection and creating the total voting outcome figure (upvote = +1 , downvote = -1). Worked on some features using bootstrap such as the footer.

Use Case Diagram

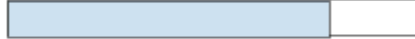


Wireframes

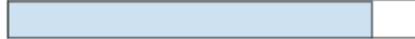


Add a review for (CLASS NAME)

Statistic 1



Statistic 2



Statistic 3



Add a comment...

POST

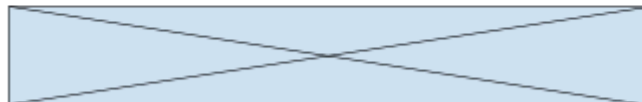
About you

Your name

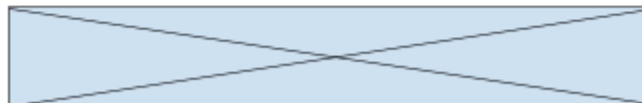
Joined may 2024

reviews posted

Other stats

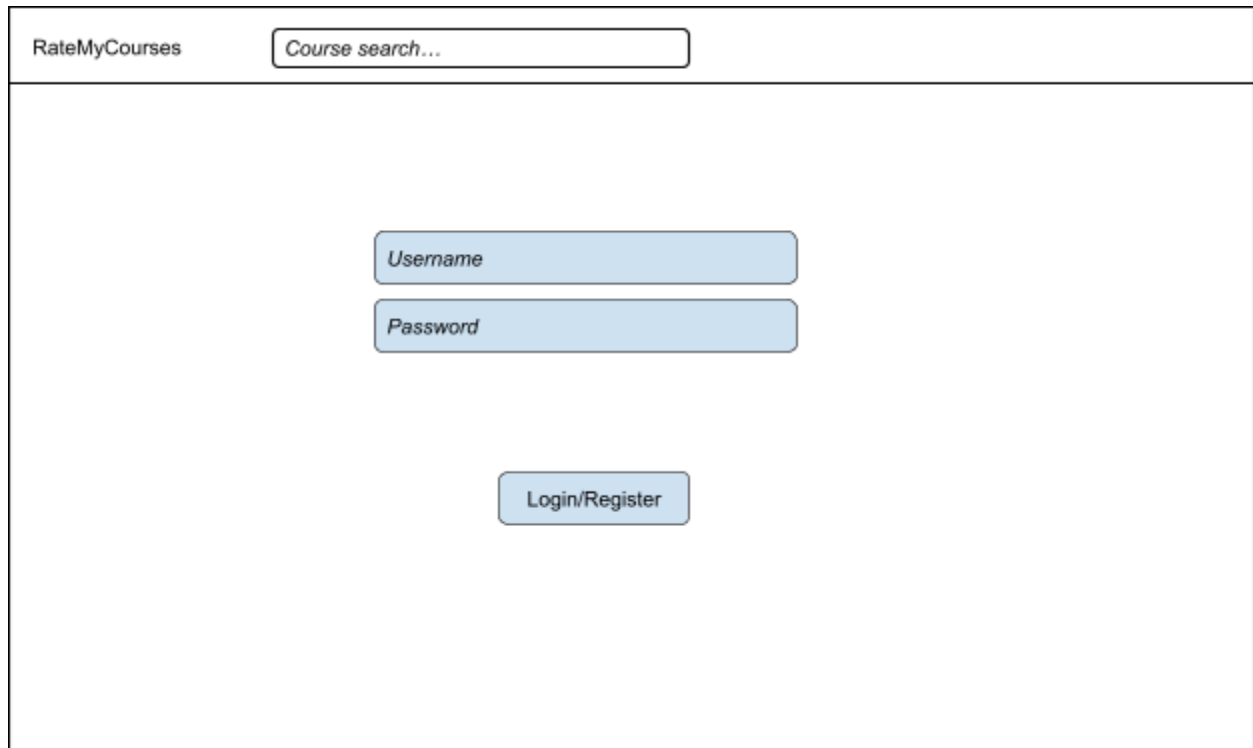
Your reviews

Edit button

Delete
button

Edit button

Delete
button



The image shows a web form for 'RateMyCourses'. At the top left is the text 'RateMyCourses'. To its right is a search bar with the placeholder text 'Course search...'. Below these, centered on the page, are two input fields: the top one is labeled 'Username' and the bottom one is labeled 'Password'. Below the password field is a button labeled 'Login/Register'.

Test Results

Find the “Computer Science 1: Starting Computing” course case:

Start Conditions:

User starts on the course search page logged in.

Prompt:

“Go to the Computer Science 1: Starting Computing course page.”

Ideal steps taken by user:

- The user clicks on the search bar.
- The user types “Computer Science 1: Starting Computing” or some substring of it into the search bar.
- User clicks on “CSCI 1300 Computer Science 1: Starting Computing”.
- User recognizes that they are on the correct page.

Results:

The users were able to accurately identify how to search and locate the class we told them to. They also knew that they had completed the task. However, many of them noticed that sometimes classes with the same name would show up in the recommended searches.

Changes Made from Results:

Matt removed the duplicate results from showing up on search.

Write review case:

Start Conditions:

User starts on the CSCI 1300 page logged in.

Prompt:

"Write a review for this course and describe what each metric means as you fill the form out."

Ideal steps taken by user:

- User clicks on the "Write a review" button.
- User successfully describes what each metric means.
- User successfully fills out the form and posts it.

Results:

The users were able to write a review only when logged in but not all of the courses could be selected for the review and the default course remained CSCI 1000 regardless of what course page the user came from. The user was able to fill in all of the information they wanted.

Changes Made from Results:

Maddie changed the default course to be based on the course page the user came from and added all courses in to allow the user to write a review for the correct course.

Delete your review case:

Start Conditions:

User starts at the home page logged in

Prompt:

"Delete the review you just made"

Ideal steps taken by user:

- User clicks on "Account"
- User clicks delete on review they made.

Results:

The review did not delete and crashed the website with the error message “could not GET deleteReview”.

Changes Made from Results:

Madison changed the form method to be ‘post’ and changed some of the parameters to ensure that they were not empty so that the correct review would be deleted.

Explain what each of the sort options does:

Prompts:

“What do you think sorting by “Trust” means?”

“What do you think sorting by “Positive” means?”

“What do you think sorting by “Negative” means?”

“What do you think sorting by “Year” means?”

Results:

Most users didn't understand what each word meant specifically. They were especially confused by “Trust”.

Changes Made from Results:

We changed “Trust” to “Popularity”, “Positive” to “Highest Rating”, and “Negative” to “Lowest Rating”.

Full website test case:

Start Conditions:

User starts on course search page logged out.

Prompt:

“Write a review for Computer Science 2: Data Structures, downvote every other review for that course and upvote the review you just wrote. Then delete the review you just wrote.”

Ideal steps taken by user:

- User searches for a course.
- User tries to write a review but is redirected to login.
- User does not have an account so they register.
- User searches for a course.
- User writes review for course
- User downvotes all of the other reviews.
- User upvotes their own review.
- User deletes review just made.

Results:

The website successfully ran and all functions were successful. Some of the redirects were not ideal according to the tester but they were able to navigate to each of the pages and perform the tasks.

Changes Made from Results:

Our login/registration pages now route to the last page you were at before logging in making the process less tedious.

Deployment

1. Clone the repository
2. Setup .env file
3. Change directory: `~/CSCI3308-Final-Project/ProjectSourceCode$`
4. Type 'docker-compose up -d' in the terminal
5. Visit localhost:3000 on your favorite browser!