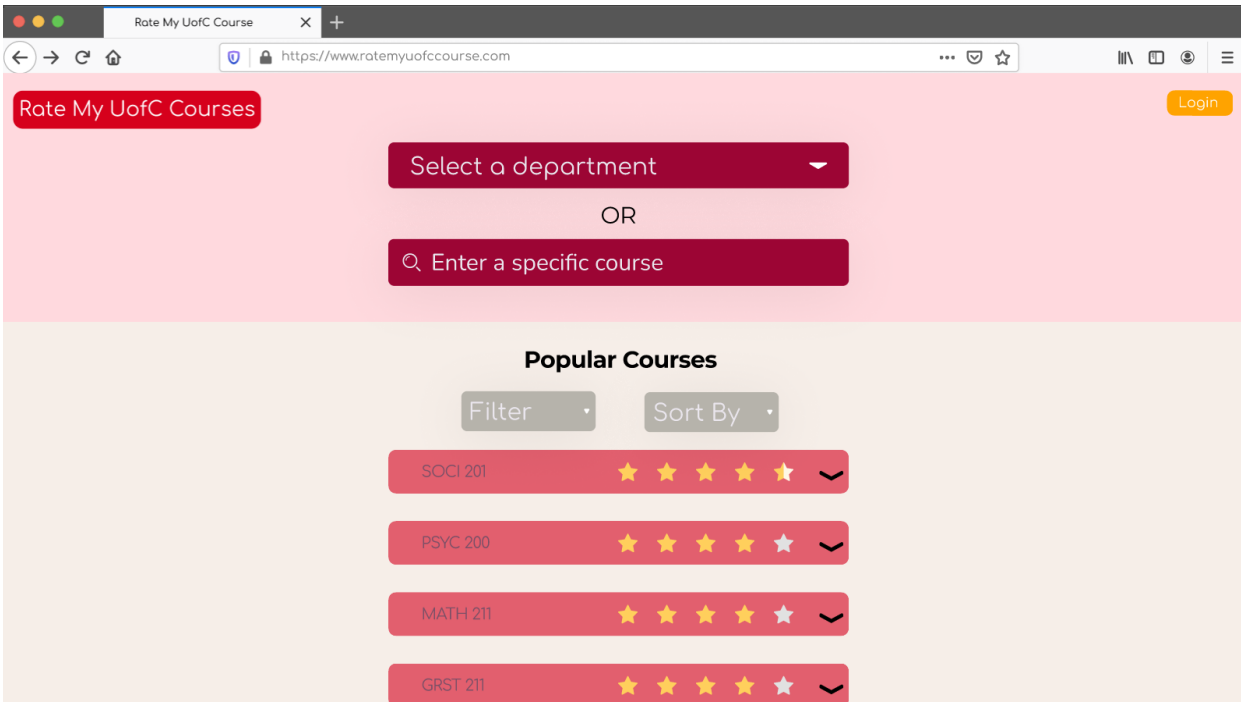
Rate My UofC Course

Project Report



Eric Tan

Introduction

The main objective for this project **"Rate My UofC Course"** is to create a website where students from the University of Calgary could get information about potential courses that they may enroll in. The information that the university provides about a course is commonly a generic sentence or two that doesn't provide the students with much information about what they are learning and getting out of the course. Our project strives to create a centralized service where students can reliably get information about courses from other students using our application. This application allows students to search for courses that they are interested in taking and they would get the ability to view other students' feedback and average rating of the course. Analogously, students have the ability to find the courses that they have taken and they can provide a rating (from 1 to 5 for the course) and provide comments about their experience and takeaways from the course.

The remainder of this report will provide *background information* about why we created the project, the *goals* that we have for this project, our *accomplishments* thus far, *functionalities that users are able to perform*, *a guide on how to perform user functionalities*, the *changes from the mockups to the final implementation*, a list of *project requirements*, a list of *technologies we used* to complete the project, *what we would add onto this project* (given more time), what some of our *general takeaways* from working on this project were and our *concluding thoughts*.

Background

Students at the University Of Calgary often have questions about what courses they want to enroll in, for many different reasons such as how informative the course is, how fun the course may be, how difficult the course is, and many other reasons. The current problem is, the university doesn't provide much information about courses beyond a short vague description about what is taught in the course, this proves to be unhelpful for students when they need to choose courses to register in. Currently students can get more information about courses from external third-party services such as Reddit and Discord to get information about courses, however the information is scattered/disorganized and the information provided may be untrustworthy. Students can also ask friends/classmates that may have taken the course previously about their experience, but the problem is that not everyone they know has taken the course(s) that the student wants to learn more about.

Our motivation for the project came from our own difficulties with finding dependable and specific feedback & information regarding courses of interest.

Our solution to this problem is a web-based application that allows students to find the courses they have taken, and to provide an informational review and rating about the course. This application aims to be an organized page where students can search for courses and get accurate information about them with ease. This can be a very helpful resource for students to learn and decide if they want to take a certain course or not; lots of different students would be able to provide information and share their experience with the course, which can help students decide whether they should take a course or not.

The primary users of the web-application would be students at the University of Calgary who want to receive more information about potential courses they may enroll in and those who want to provide information about courses to help other students. The main demographic of these users are adults within the age from 18 to 30, so it is very likely that they know how to navigate through different web-applications, but we intended on making the design as simple as possible for any user with any experience of technology to use.

The other users of our system would be moderators (who may be students as well); they would overlook and maintain the system. Their main task is to maintain the system by making sure students are posting appropriate/relevant comments and making sure the system has an updated list of provided courses from the university.

Project Goals

The main goal of this project is to create a usable web application for students at the University of Calgary to publicly view information about courses and to rate courses offered by the university. During the creation and development of the project we identified further goals.

- **Process Improvement -** Our goal is to improve the current method students use to gain insight into courses.
- **User experience -** Our goal is to create a good/satisfactory user experience for users of our web application. We did this by incorporating intuitive and modern components into our design; this ensured users had little to no learning time using our web application.
- **Reliability -** Our goal is to ensure that the information shared on our application could be reliable. We did this by adding in moderators of the system who are responsible for

monitoring the reviews added are appropriate and relevant, while also keeping the courses on the system up to date with what is actually provided by the university.

Project Accomplishments

Users are able to find the courses they are interested in, read what others have posted about the course and view others' ratings, and the ability for users to rate and review courses they have taken. Users can use the system anonymously (not logged in), and if logged in (after registering for an account), they gain the ability to view their posted comments and to delete their comments.

We also have a moderator role that can add, modify, and delete courses and departments on the system. They can also remove comments posted by users if it is considered spam or inappropriate.

As a concrete example, a student who is interested in the course "SENG 513" can use the system and search for the course, and once they see it displayed on the screen, they can expand it to get more information about the course. Here they can view the description of the course (provided by the university), what the average rating of the course is, and individual comments about the course (provided by other students). Students have an option to click the "add a review" button where they can provide their own feedback and rating about the course. This is the primary and most common task that users will perform when using this system.

Project Description

To run the website:

1. Open the project folder in the terminal. Sample path: "yourProjectDownloadPath"/SENG-513-Group-Project where "yourProjectDownloadPath" is the path from where you downloaded the file.
2. Install the required node modules by entering `npm install` into the terminal.
3. Enter `node server.js` to run the server.

4. On your preferred browser, go to http://localhost:3000/.

To access Admin/Moderator Login for additional features:

To test the admin/moderator, please use one of the following credentials for logging in:

Email	Password
admin@admin.com	abc
bob@admin.com	abc
pam@admin.com	abc

Existing Student Users Login:

To test a pre-existing user, please use one of the following credentials for logging in:

Email	Password
et@test.com	abc
user@user.com	abc
sam@ucalgary.ca	abc

Users would be able to find the courses they are interested in. They can accomplish this by either searching for the course directly using the search bar, finding the course through the department that they selected, find the course in the default “popular courses” list, or they can find courses using filter/sort by options. *See Appendix Search, Filter & Sort*

Users are able to rate and review courses. Users would first navigate to a course (by one of the methods previously listed), expand it, and they would get the option to use the “add a review” button. The user will get redirected to a page where they can rate the course from 1 to 5 stars, write their comments on the course in a text area, and add some predefined keywords that they feel describes their experience with the course. After expanding the course, Users would also be able to view a course description (provided by the university), the ratings and comments about that course (posted by other users), reply to the posted comments, and there is an upvote/downvote feature for if the users found this comment to be helpful/informative (aka, rating the comments). *See Appendix Expanded Course, Review Page*

Users are able to perform the main features of the system either anonymously, or when logged in (after a user has registered for an account by entering their email and desired password). If the user is logged in, they get the ability to view the comments they have posted (in a list format) and to delete their comments; both features are accessible through a private user profile page. Users can delete their comments by either expanding on the particular comment they want to delete (and clicking the trash button), or they can delete multiple comments at once by clicking the “delete comments” button on the user profile page, which will allow them to select the comments to delete by clicking the white box.
See Appendix Register, Login and Profile

For our moderator users, they have the ability to add, modify, and delete courses to/from the system and the ability to add departments to the system (to keep the application accurate with the university information). Moderators can also delete comments if they consider it spam/ inappropriate. Moderators can access these features through the drop down button seen on the main screen of the system. The add and modify courses pages for the moderator are simple pages that just require the course name and description, where the description can be modified. The admin delete courses feature allows the moderator to search through all the courses (the same way students can) and they would be able to select the courses to delete by checking the white boxes beside the courses. *See Appendix Dropdown Menu, Delete Courses, Add Course, Modify Course and Add Department*

Mockups vs Final Project

There are no major changes to the system design/interface when comparing the prototype/ mockups to the final project implementation. There may be some minor styling differences due to not being able to directly use the same styling from the figma prototype on the final implementation and some formatting/position changes were made to make the system look more pleasing to view.

The major change from the mockup stage to the final project is the implementation of logic to our system. During our prototyping stage, we added some dummy functionality such as what happens to the system after a button is clicked or when input is entered, but we didn’t apply any logic/backend programming to this component, since it is just the prototyping stage. On the final

project, we implemented all the logic so the system works like an actual system. For example, in the prototype, when we click the department search, it displays a few departments that we explicitly created for prototyping reasons (ie. the resulting list was manually created by us), but in the final project implementation, this feature is achieved by navigating through the mongo database, selecting the departments from the database, and creating them as list elements for users to view and select. *See Appendix UI Mockup & Final Page*

Project Requirements

Proposed Project Requirements:	**Implementation Explanation**:
Search for existing courses.	This has been implemented. Users are able to navigate to a search bar where they can enter the course they are interested in and the page will display that course.
Search for departments that provide courses	Our project meets this requirement because users are able to find the department they want to through a drop-down list containing all the departments. There can be visual improvements to this as it is using a generic drop-down list view.
Search ascending/descending ratings of courses	There are several sorting options on the main page that the students can use to view the courses in their preferred order, including view by ascending/descending ratings of courses.
Add/remove courses to the system.	The moderator has the ability to add courses to the system. Once the moderator is logged in, they have access to a drop down that provides a lot of moderator options; one of them leads to a page where the moderator can enter the course title and description of the course to the system.

Add/remove departments to the system.	This has been partially completed. We were able to add departments to the system by either explicitly adding new departments (using the moderator feature “Add Department”) or implicitly adding new departments (using the moderator feature “Add Course” for courses to be added and the department doesn’t exist in the system). However, we were not able to implement the remove department feature. As we thought about implementing it, we realized it is a really niche feature because in reality departments at the University typically do not get closed, thus no need to be removed from the system.
To submit/delete new ratings (from 1 to 5 stars) for specific courses.	Yes. Students can expand the course and get the option to add a review, where they can choose what ranking from 1 to 5 they want to give the course by hovering over the stars and choosing their rating of the review page.
To submit/delete new comments/advice about a specific course.	Students can expand the course and get the option to add a review, where they can successfully enter a written comment about that course that they have taken that other students can view. However, the ability for students to delete their own comments did not get implemented due to time constraints.
Vote on reviews.	Students, whether or not they are logged-in, can upvote or downvote another student’s review by clicking the arrow buttons on the right side of the review.
Give feedback/reply to existing comments.	Not implemented. Although we have the buttons for them in the reviews, it only shows dummy data when “Show Replies” is clicked. This proved to be a problem as we tried nesting comments with each other, and thus led to scrapping this functionality.

Change the details of existing courses.	Moderators are able to look for a course and change the description of the course. They can do this by entering a course that exists inside the system in the course name field, and then when they click enter, the current description of the course will be populated, and thus can be manipulated
An option to login/sign out of the service.	Both the students and the moderators can login to the system using the same login button, and both can log out; moderators access the logout/signout button through the moderator dropdown features, while students can sign by navigating to their user profile page.
For moderators, the ability to delete/hide any comments that are not informative/ inaccurate.	Partially met. The moderators are able to completely delete reviews by selecting a course, viewing reviews, and selecting the “Delete Rating” button, but not hide/show them.

Main Technology Used

- **HTML** - We used HTML to build and structure the content of our website. We used this as it is the building block of creating web pages.
- **CSS** - We used CSS to design the web pages we created. We used CSS as our prototype was created in Figma and it produced some of the CSS code needed for our design. We also manually wrote our own CSS/SCSS code where required, for example, positioning of elements.
- **JavaScript** - We used JavaScript to allow for user interaction and to apply logic to the website's functionality. Each page of the program utilizes some JavaScript.
- **Express.js** - We used Express.js to connect the client and database using REST API. For example, if the user wanted to add a review, we used Express.js to listen for requests, then send the POST request to the database (MongoDB).
- **Node.js** - We used Node.js as it was a necessary tool for sending client side requests and implementing mongodb, and other node modules.
- **MongoDB/Mongoose** - We used MongoDB to store and save user ratings and comments, store course and department information, and to store user credentials.

Mentioned under *Project Description* and the README file, after downloading and installing NodeJS, all node modules required for the project can be installed by entering `npm install` in the terminal. There are some node modules that were used for a small portion of the system, such as bcryptjs for storing user entered passwords as hash values in the mongo database.

Future Work

Polishing up the front-end to obtain a more modern look compared to popular web-applications. While we are happy with the visual components of our application, if we had more time to work on this, giving the system a visual look similar to popular applications like Twitter, YouTube, Netflix etc., would be the best case. We were not able to achieve this “modern look” because most of our time was dedicated to implementing the features of the system, and it wouldn’t make sense of using our precious time to make our already satisfying visuals even better.

Security: While not a necessary feature for this project, we did not really consider any security features at all when implementing the project. Initially, we intended to store the user registered password in plaintext in the mongo database, but we later encrypted it due to a non-security related issue. There are also no restrictions on the password the user decides to use; the user can choose whatever password they want to use, regardless of how weak it is. There can easily be password restrictions implemented, such as minimum number of characters, usage of non-alphanumeric symbols, etc, but this was something that we overlooked and didn’t realize until we were almost out of time.

Hosting the application on the internet. While it isn’t the most difficult or time consuming thing to do, if we were able to add a few more features and to polish up the system a little more (with some of the things we have mentioned), it may be possible for this to be used by real students from the University of Calgary.

An automated way to check for spam. Right now, we have the admin/moderator manually deleting reviews, and if it was deployed to a larger user base, the moderator would have to check comments for numerous courses and comments, which is obviously tedious. If we had more time to work on a spam-detection algorithm, this can fix the problem, but something like this will take a lot to implement.

A complete database filled with all the courses and departments at the university. The admin/moderator needs to manually add or delete courses on our website. We initially wanted to script all the courses provided from the university into the database, but we were not able to access a document containing all the courses and departments. Since we had to manually add courses and departments to the system, it is a time-consuming process, so we obviously couldn’t add it all and only added in a small subset of courses and departments.

Finishing our missing functionalities. Due to time constraints and technical difficulties during implementation, we were not able to implement replies to comments/feedback, and the ability for users to delete their own comments. If we had the time, this would be something that we would like to implement.

Lessons Learned

During the prototyping stage, our team worked with Figma. In doing so we were able to collaborate with each other in real time. This was really valuable in the creation of our prototype as it helped us save time and avoid confusions & conflicts. Had we decided to split up the prototype and work on it in separate files it would've been messy and time consuming to compile.

Password encryption. We did not intend to focus at all on how passwords were stored and encrypted initially; we were just going to keep passwords in plain text in the mongo database. But we had a problem with maintaining login state when using socket.io, so when we switched express, we realized that it was easier to implement if we had our passwords encrypted, so we decided to use bcryptjs to help encrypt the passwords that are stored in the mongo database.

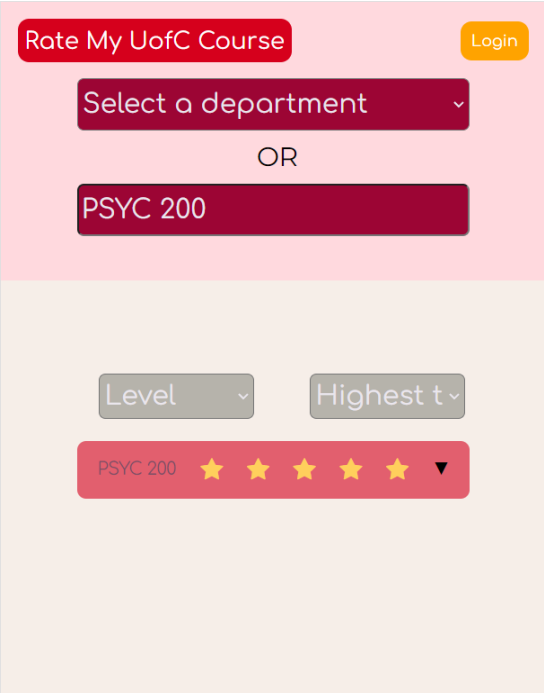
Initially we were using socket.io to connect the client to the server, but we had difficulties keeping track of user sessions because socket.io is not persistent. This meant that whenever a user is logged in and they navigate to a new page, their login status would be gone, and they would have to log back in again. We switched to express.js to solve the user session/state, which fixed the issue of tracking user sessions/state as they navigate throughout the application.

Conclusion and Miscellaneous

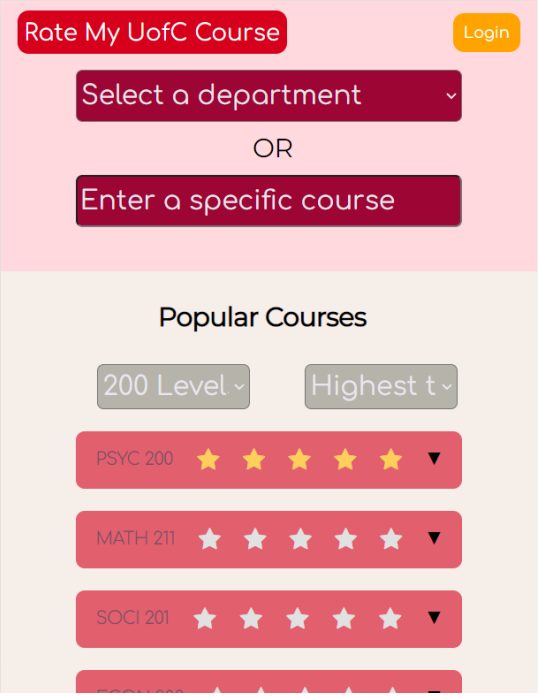
Overall, our application “Rate My UofC Course” is a web-based system where students from the University of Calgary are able to write comments and provide ratings about a course that they have taken so other students can view them and decide whether they would be interested in enrolling in that course. This application is a reliable, centralized, and simple solution that students can use to deal with their problem of wondering if a course they may want to register in is right for them. With some more polish on some of the lower-level features, this application can

come very close to being an actual application that can be publicly used by many students at the University of Calgary!

Appendix

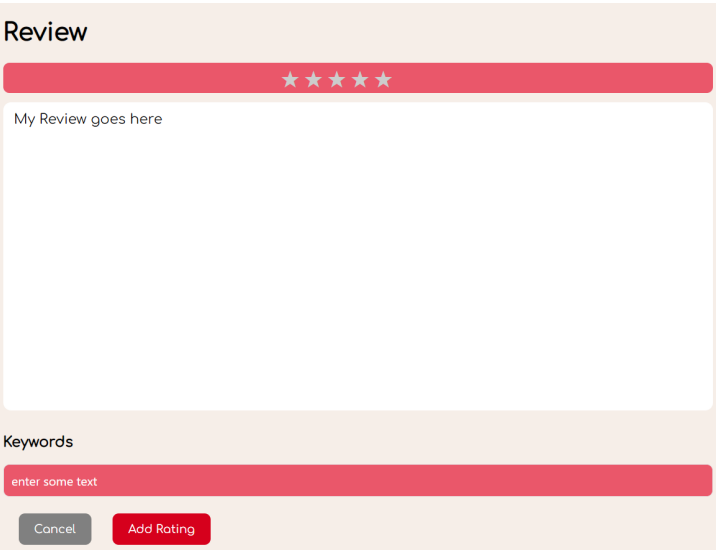


Search

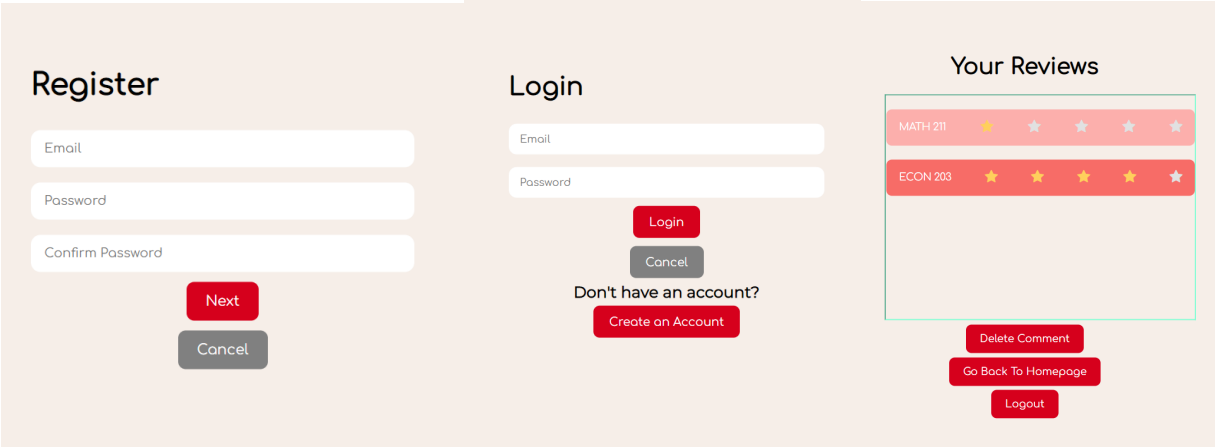


Filter & Sort

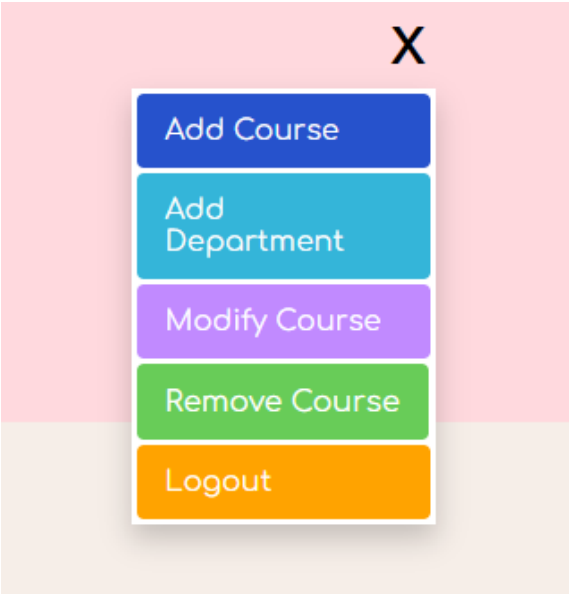
Expanded Course

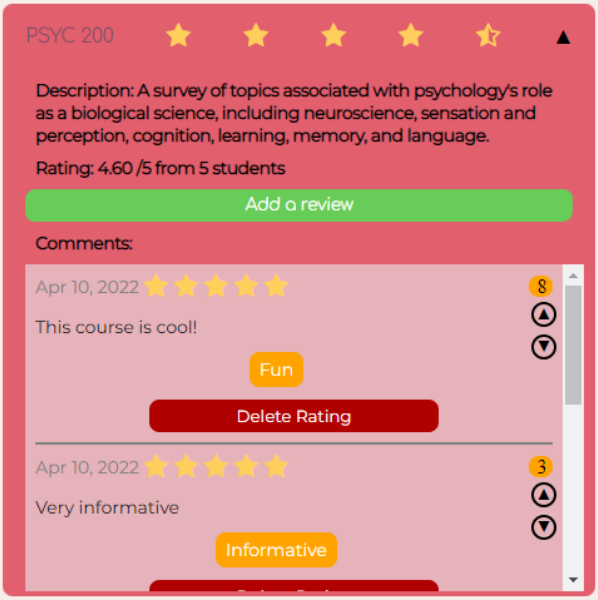


Review Page

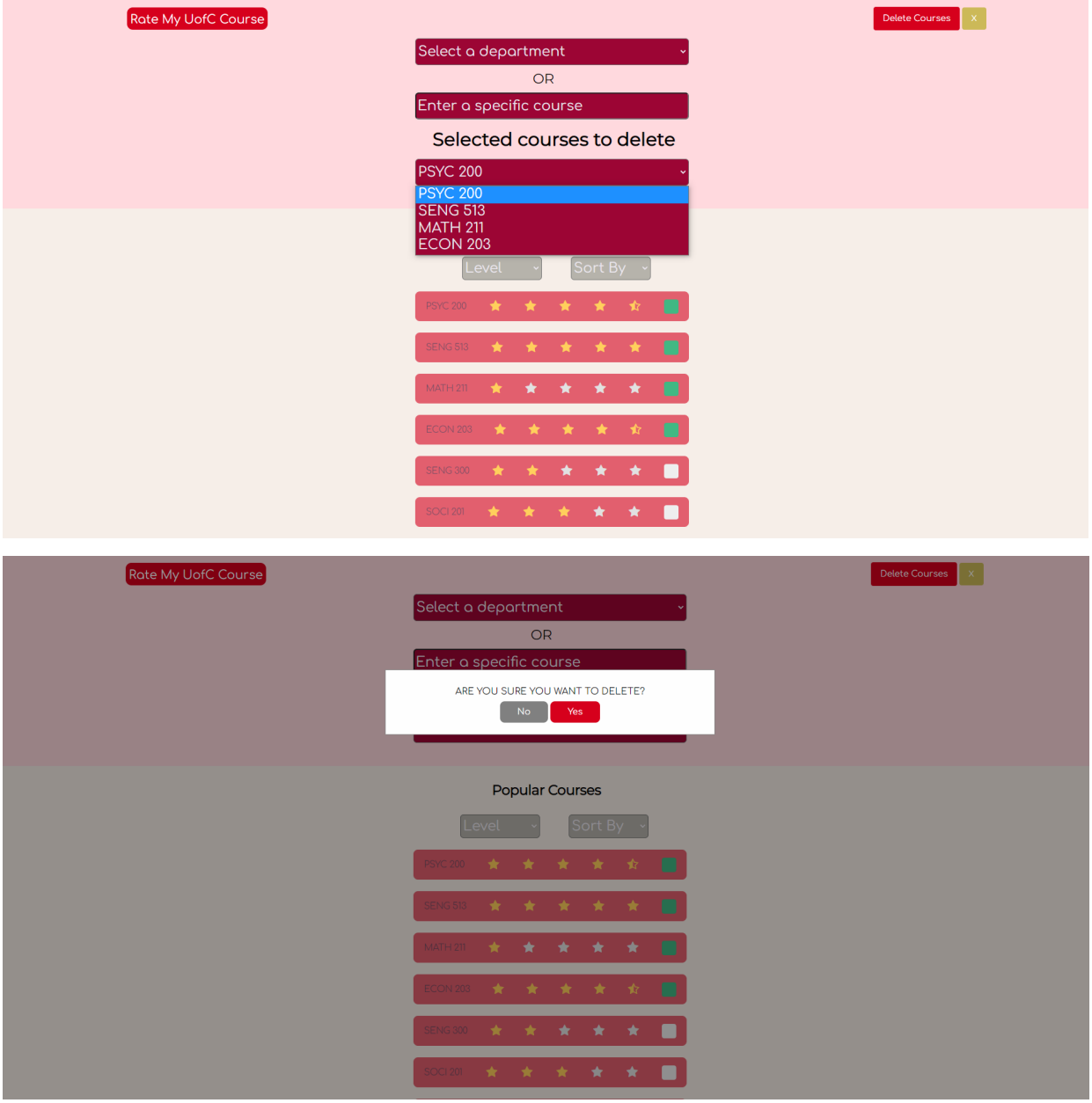


Register Login Profile

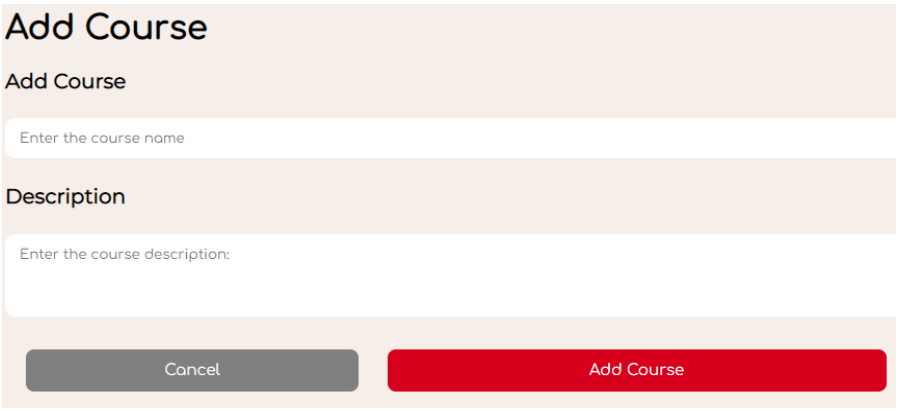




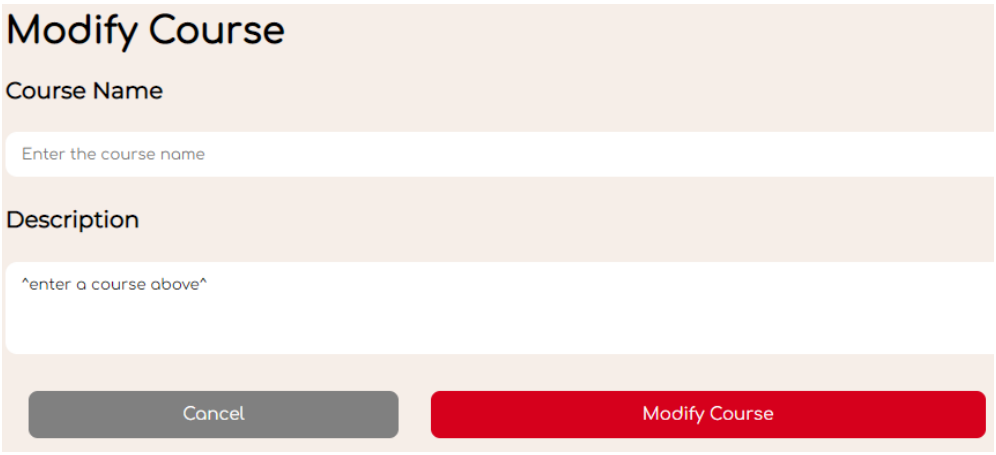
Drop Down Menu Delete Rating



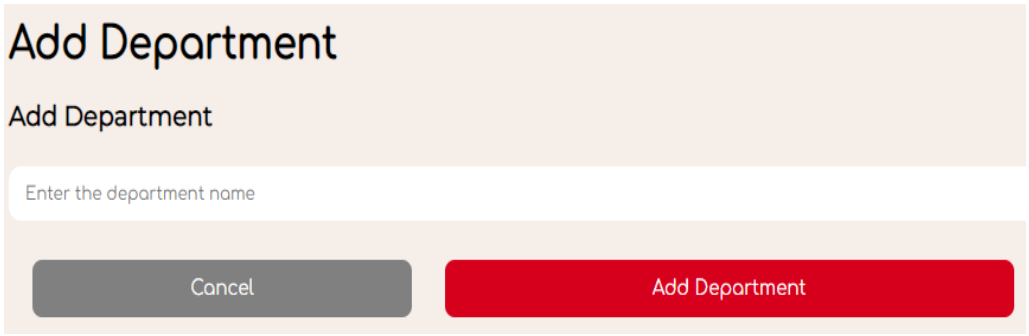
Delete Courses



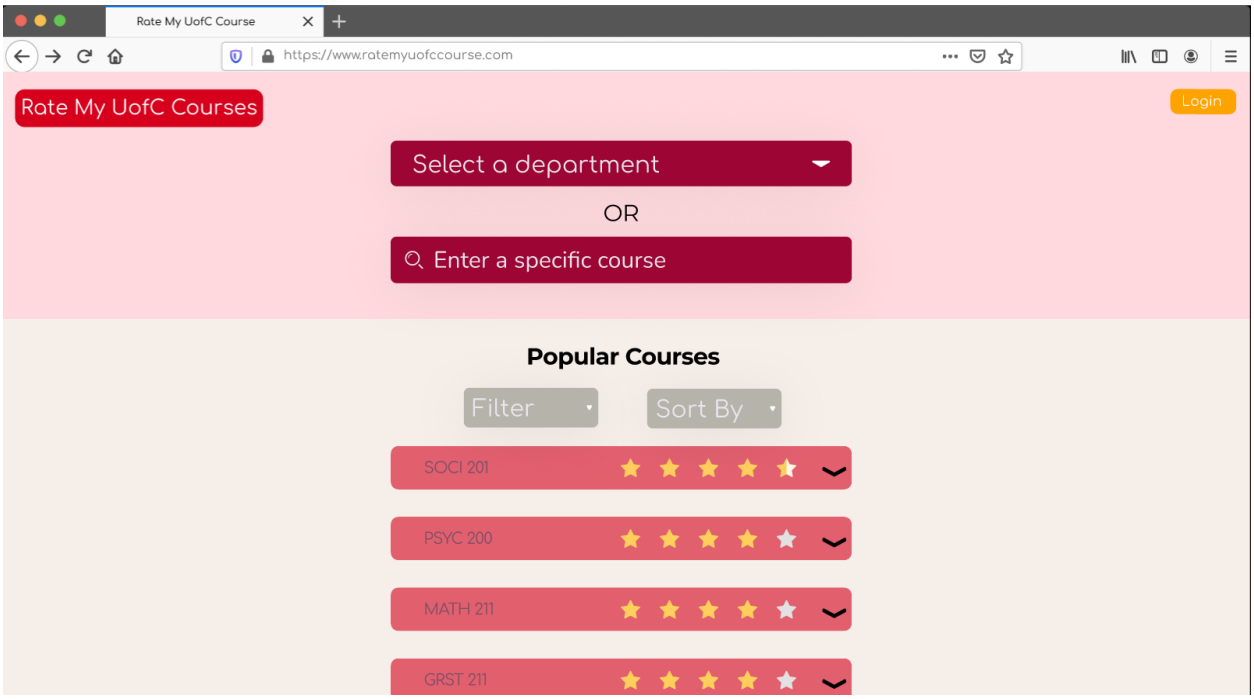
Add Course



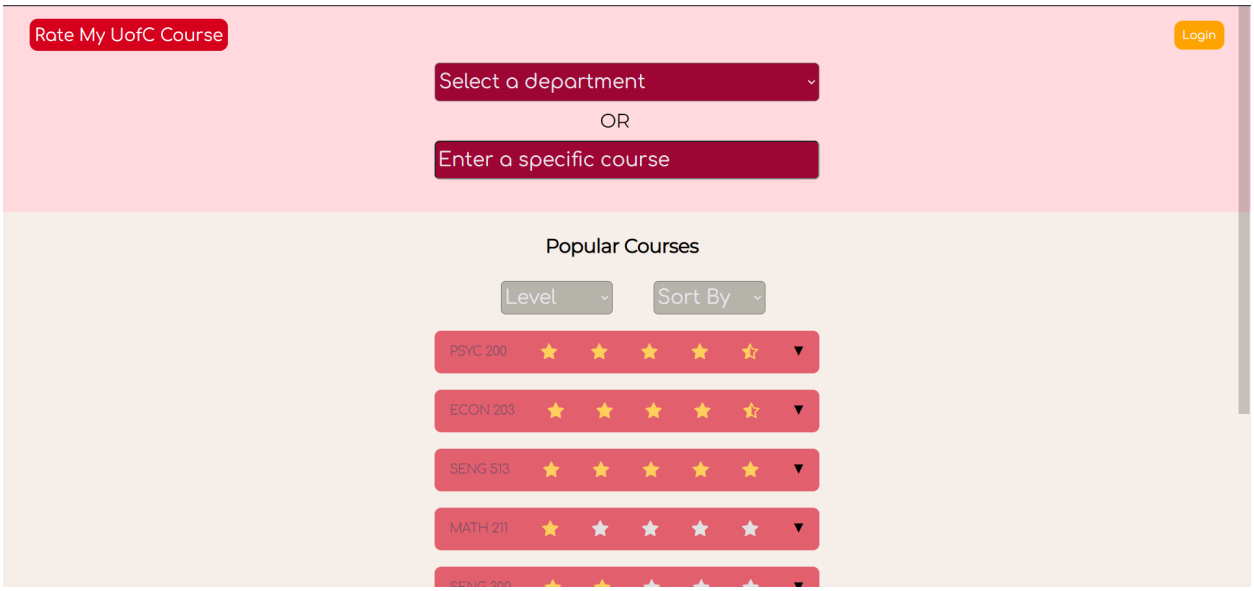
Modify Course



Add Department



UI Mockup



Final Page