Replicating the Entire Product Development Process: Building a Better Alternative to Ratemyprofessors.com

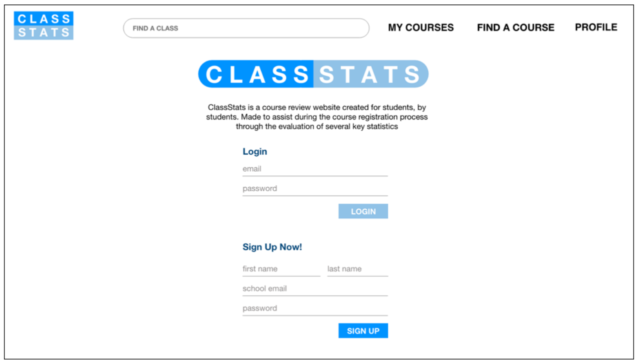
Julian Chan
Senior Comps Final Paper: Computer Science
Occidental College
`mchan2@oxy.edu`

Abstract

This paper explores the end-to-end process of modern day product development. Criteria of the optimal website is discovered through the analysis of existing research and is applied to the development of a REST API, SQL web application. Specifically, this paper details the step-by-step process in creating a fully functional, more user experience-centric, alternative to course review websites such as Ratemyprofessors.com. From user research and market validation, to UI/UX design, to development, each stage in the product development process is replicated and documented. Some of the web development tools utilized for this project include Node.js, React.js, and MySQL along with other frameworks such as Bootstrap. Explanations on why specific frameworks and tools were selected is also provided. This research and documentation should interest those intrigued by web development, UI/UX design, or the overall process on how modern day tech applications are developed.

1 Introduction

The average American spends 24 hours a week on the internet. That's 1248 hours a year spent staring at a blue screen scrolling through webpage after webpage. As of 2019 there is over an estimated one billion websites online, and yet most people spend their browsing efforts on only a select few. Why do most people use YouTube instead of Vimeo when they try to stream online videos? Why do most people resort to Google instead of Yahoo when they want to do a quick search? These are the questions that fueled the inspiration to delve into the world of web development. Specifically, the goal for this project was to replicate the entire end-to-end product development process from ideation, to design, to development of a fully-functional REST API, SQL server that serves as a better alternative to Ratemyprofessors.com. This website, named ClassStats, also attempts to embody all of the characteristics of the "best" possible website. These characteristics include fast loading time, quick implementation, beautiful user interface, carefully curated user experience, and responsive capabilities. A course review type website was chosen for this project because of its relevance to college students. Every semester during course registration, students scour the pages of Ratemyprofessors.com in hopes to find some bit of information that will aid in their decision to enroll in a specific class or not. However, Ratemyprofessors.com lacks in many aspects. The first being the quantity of course reviews it provides. Ratemyprofessors.com has been active since 1999, garners an enormous 1.7 million professors, eight thousand schools, and boasts an impressive six million monthly visitors. And yet, most students find that the professors they search for, some of which have been teaching well over 10 years, have little to no reviews to their name. The math doesn't add up. In addition, a second aspect that Ratemyprofessors.com is lacking in, is its inability to produce high-quality, valid reviews. The issue here is that everyone can submit a review on Ratemyprofessors.com, the website does not enforce any form of validation check to see if the user writing the review actually took one of that professor's classes. It calls into question the robustness of the data provided on the course review site. And so the motivation for this project was not just because course review websites are extremely relevant to students, but also because it has so much more potential than its current alternative and improving upon it will only more efficiently assist students during the course registration process. Moreover, the scope of this project was one that required the embodiment of each role in a typical product development team, being able to learn how a UI/UX designer and developer problem solves in different points during a product lifecycle will only deepen the knowledge of anyone interested in how modern day products are actually built.



ClassStats is a course review website created for students, by students. Made to assist during the course registration process through the evaluation of several key statistics

Figure 1: ClassStats Login/Signup

2 Background

Living in the 21st century, we are constantly bombarded by various forms of online media. People of all ages and backgrounds are consistently surfing social media platforms, scrolling through online news sources, or viewing their favorite movies and TV shows on streaming websites. We are interacting with the internet and its web pages repeatedly and increasingly. And yet, despite the many beautiful, functional, user-friendly websites out on the world wide web, there is still a vast majority that are just plain awful. Some of these websites have faulty loading issues, an inability to process client requests quickly, a lack of adaptability to different sized devices, a subpar user experience, or are just flat-out ugly. There are so many different factors that play a role in making sure that a visitor to a website is engaged and satisfied. As explored in "Relating Experience Goals with Visual User Interface Design" by Jussi P.P. Jokinen, Johanna Silvennoinen, and Tuomo Kujala, first impressions matter. "They (participants) indeed made their appraisals on the basis of first impressions" (Jokinen, 387). The smallest subtlety could instantly trigger a subconscious response which could cause users to immediately leave the website upon entering. In order to ensure that this doesn't happen, all of these factors must be taken into account well before the site is deployed onto the internet. There is a huge variety of different processes and factors to account for in the development of a website, which is why most development teams assign multiple engineers and designers different responsibilities in order to make certain the final product is outstanding. These roles include a front-end developer, back-end developer, and a UI/UX designer. The individual responsibilities behind each of these positions are extensive so most teams will actually consist of more than just these three roles. It is the UI/UX designer's job to study the interactions between the product and user in order to design and create a mockup of the website before any of the actual coding takes place. "Designers should emphasize more on the experiential aspect of the interactions between users and products, so as to understand potential UX and to implement design for experience" (Zheng, 3). The developers then use code to take the UI/UX designer's wireframes and bring them to life, as well as implement different API's that will help with the overall functionality of the website. The front-end developer collaborates with the back-end developer to make sure that the front-end is integrated well with the database. In reality, these three positions possess far more tasks than what has been previously mentioned, but their roles do center around these general responsibilities.

Some of the factors that contribute to a successful website is a beautiful UI or user-interface, a carefully thought-out UX or user-experience, speedy processing of the webpage's content, quick and easy data queries, and a responsive capability to adapt to different sized windows such as tablets and smartphones. Aside from the UI/UX design of a website, all of the problems that come with the functionality of a webpage can only be fixed by the developers. And in fact, some of the solutions that will be presented not only remedy these issues, but also make the jobs of these developers significantly easier. Most of these solutions are not necessarily brand new ideas, but instead are tools, concepts, or frameworks that are widely accepted by most developers in the industry today. These solutions will further educate those who are looking into designing and building their own website, or provide them with enough knowledge to know what to look for in building a web development team.

2.0.1 UI/UX

One of the first steps to making a good website is to conduct proper, professional UI/UX design. Most people use the terms UI and UX interchangeably when talking about a website's visual aesthetic. However, the two terms actually refer to two separate concepts of web design. User interface (UI) is centered around the structuring of a webpage and how it looks, whereas user experience (UX) is focused on the actual journey through the website.

2.0.2 User Interface Design (UI)

User Interface design is how certain colors, icons, font-sizes, image positioning, and other small details are placed on a website in order to influence the user's perception of the site's content. These small factors actually trigger significant psychological responses in the human brain. "Color and screen design directions have various psychological and social associations" (Cyr, 3). For example, a psychological association could refer to the association of the color red with danger. This is why most delete/exit buttons and warning messages are presented with the color red, as it represents a destructive property such as wiping something from memory or exiting out of a page. "The fact that red is strongly associated with danger has led to the widespread use of red icons and signs to indicate hazards" (Hsieh, 740). And so if these features possessed a color like green, it would elicit a conflicting response within users as green is not typically used to describe danger-associated actions. Tsuei-Ju Hsieh of the Department of Information Communication in the University of Taipei, Taiwan, found that "warning icons could be detected more quickly when colored red and blue; however, most viewers favor red icons" (Hsieh, 741). Hsieh also found that certain colors influence users' perceptions of different icons and their meanings. In Hsieh's experiment, a group of 96 participants were recruited from a course in experimental design in the Chinese Culture University. Each participant had experience using computerized devices and had used them for roughly one hour a day for a period of five years, every participant also had normal or corrected-to-normal vision. The participants were asked to evaluate their opinions of certain icons via survey. These icons were all altered in some way, characteristics such as the original color of the icon, a grayscale image of the icon, and the inclusion of a color complimentary to the original icon, were manipulated for the participants. After the experiment, Hsieh found that certain colors for certain icons elicit a much faster response or recognition within users as opposed to other colors. "Our results confirm that correct color information is crucial to naming accuracy and the speed at which icons are recognized" (Hsieh, 750). So not only do these small subtleties such as color make a website look visually appealing, but also they enable users to make quicker decisions due to the associations certain colors have with certain actions/meanings. Additionally, several other factors also encourage users to make quicker decisions and can even direct their attention to certain parts of the screen. Skilled and experienced UI designers will make it so that certain elements influence the user to look at specific, more important parts of the webpage. For instance, in an ecommerce website, the UI designer will position the icons, sub headings, and other features in a way that nudges the user to focus on the products the website is selling. "The use of user-interface design elements is to guide people's behavior in digital choice environments" (Schneider, 68). The entire goal for the UI designer is to subtly influence the user's decisions and choices as they move about the webpage. "Designers of choice environments, or 'choice architects,' can thus use these heuristics and biases to manipulate the choice environment to subtly guide users' behavior by gently 'nudging' them toward certain choices" (Schneider, 67). This idea of visual components evoking certain behavioral responses is explored in extreme detail in Katerina Tzafilkou's paper "Diagnosing User Perception and Acceptance using Eye Tracking in Web-Based End-User Development." Tzafilkou's goal in her research is to both explore the experimentation and validity of data compiled from eye tracking software, as well as fundamentally prove the idea that certain visual features of a well-designed website capture the user's attention long enough to encourage an action. In her experiment, Tzafilkou presented a participant pool with several different stimuli, among which included different types of web pages, and recorded their eye movements in relation to their actions on the computer screen. Tzafilkou actually found that the perceived ease of use of a functionality was related to the number of eye fixations the user had, which eventually turned into clicks. "Results in Table 3 confirm hypothesis H3.1 since the correlation between perceived ease of use and number of fixations that turned into clicks is significant at the 0.01 level" (Tzafilkou, 33). Essentially, the easier a button's meaning on a website was perceived by the user, the more fixations they had on that part of the screen, which eventually led to them clicking on that button. Tzafilkou's research highlights the importance of a well-designed user interface as it can encourage users to perform certain actions. Thus, an average UI designer is able to make a website look amazing. However, an exceptional UI designer is also able to design the website in a way that maximizes the user's visual perception of its contents.

2.0.3 User Experience Design (UX)

User experience (UX) is what the users feel and the momentary evaluative impression they receive when interacting with a product or service (Theopilus, 3). In a web development context, it is the perception users have of a website from a usability standpoint. Factors such as ease of use, intuitive functionality, and time optimization all play a role in the user experience of a website (Theopilus, 3). Ease of use refers to how simple a website's navigation is to a new user. If a website is difficult to navigate, meaning it's hard to get from the homepage to wherever the user wants to go, it can discourage a user from ever returning to the website. Intuitive functionality refers to the perception of certain features of the website. Features such as sub headings, buttons, links etc. For example, typically when a user wants to navigate back to the homepage of a website their initial instinct is to scroll to the top of the page and click on the website's logo or the "Home" heading in the navigation bar; this is an intuitive response. However, if for some reason the aforementioned actions redirected the user somewhere unexpected or the user has to click through different parts of the website just to get back to the homepage, it could negatively affect their overall user experience. Time optimization refers to the amount of time it takes for the user to get to where they want to go or do what they want to do. If it takes more than 10 clicks for the user to purchase an item from a website, this can discourage them from returning to use the website's platform as the user experience is not time efficient. An example of a website that has an outstanding user experience and expresses effective time optimization is AirBNB. AirBNB embodies the "3 click philosophy," where users that enter their website have to perform no more than three clicks in order to book a home (Williams, 1). The user experience of a website can also be separated into three main stages, Expected UX, In-Progress UX, and Overall UX (Theopilus, 4). Expected UX refers to the user experience before they even begin to use the product or service and is typically influenced by factors such as past experiences, comments from peers, and reviews from media. In-progress UX refers to the perceived user experience when a user actually interacts with the product or service. And finally, Overall UX refers to the lasting impression the user is left with having interacted with the product or service (Theopilus, 4). User experience does not just refer to the user's interaction with the website, the entire user-product relationship encompasses factors outside the two entities. "The influence on UX is comprised of the components of a user-product interaction and its surroundings" (Lin, 1744). Some of these surrounding factors can include the perception of the product or service in printed media (articles, magazines), in video media (streaming platforms, ads), and in indirect accounts (peer reviews) before the user actually interacts with the product (Lin, 1744). User experience designers have the more technical responsibilities when it comes to web design. They have to do an extensive amount of market research on the website's product or service, as well as come up with ideas that maximize the user journey through the website. It is the user experience designer's job to create the wireframes for the website as well as design the flow between each web page. The UX designer also has to see how their design fairs and go through several testing and refining stages through feedback from test users. Once the design receives positive feedback, the final adjustments are made and the design is considered complete.

2.0.4 Developer Inefficiencies

With the increased usage of the internet in day to day life, users are more demanding than ever. Websites are expected to load significantly faster than in the past and developers are expected to deploy website updates on a day to day basis. The entire process and evolution of web development has been accelerated due to an upsurge in demand for functional, updated, fast websites (Aggarwal, 133). As a result, the traditional approaches to web development have become obsolete and in dire need for supplementary assistance. Although it works just fine, the old-school approach of linking multiple HTML, CSS, and JavaScript files together to create a website has become insufficient. A little background on what each of the three aforementioned programming languages do. Hypertext Markup Language, or more commonly referred to as HTML is the extreme basics of a website. HTML is typically used to compile all of the site's raw content such as text headers, buttons, images, paragraphs, links, forms, and videos into one file. "HTML has a set list of tags (such as header, anchor, paragraph, etc.). Tags denote the type of information. The Web browser uses these tags to display the text appropriately in a manner best suited for the client hardware, software, and user preferences" (Blansit, 396). Think of HTML as the skeleton of a website. Cascading Style Sheets or more commonly referred to as CSS, is responsible for the styling of the different HTML components on a website. Things such as making the header on a website the color blue, size 30 font, and positioned 150 pixels to the right are all results of CSS. Additional examples include making an image larger or smaller, or styling a button into the color green with rounded edges. Think of CSS as the flesh and exterior features to the existing HTML skeleton, it is primarily responsible for making the website look nicer by altering existing components in the HTML file. JavaScript is responsible for the functionality of the website. Specifically, how certain elements on a website interact with each other and the user. For example, making the user's name pop up on the screen after he logs in, or pre-loading the content of the website so that everything appears at the same time are all products of JavaScript. Although the traditional approach to web development is still used today, most developers are turning to specific "stacks" to meet the demands of huge enterprise web applications (Aggarwal, 132). A "stack" is a set of programming languages and frameworks a developer uses to program both the front-end and back-end of a website. A stack that is widely used today and is proven to assist developers in modern day web development is the MERN stack.

2.0.5 MERN Stack

The MERN stack refers to "an end-to-end framework for developing dynamic web-applications, starting from the top (code running in the browser) to the bottom (database)" (Aggarwal, 133). The MERN stack consists of four primary programming tools that all revolve around one central programming language, JavaScript. The four tools are MongoDB, Express.js, React.js, and Node.js. The reason why developers are turning to stacks like the MERN stack is because of its ability to create extremely fast-processing web pages, as well as the comfortability it provides as everything is centered around JavaScript concepts and methodologies. In addition, all of these tools are open-sourced, meaning that they are experiencing updates from developers on a consistent basis, thus continually expanding their usage capabilities.

React.js is a JavaScript library that enables the development of large applications which also allows webpages to load new content without constantly needing to refresh. Traditionally, whenever a user clicks on a new page, the web browser deletes the entire code of the previous page and loads a brand new one into the browser. But now, instead of wiping out the entire code of the previous page, React.js keeps the old code and only loads new code for the new elements on the new page. For example, a lot of websites have the same navigation bar at the top, this bar could display links such as home, contact us, about us, etc. So whenever a user clicks on a new page, React.js is able to keep this navigation bar at the top and only updates the elements of the webpage that are different from the previous. This results in much faster websites. React.js is also used to develop reusable user interface components through the building of modular pieces of code (Aggarwal, 133). Not only does the React.js library assist the experience of the website on the client side, but it also expedites the development process by allowing the reuse of certain pieces of code, thus making the lives of developers much easier. Instead of having to rewrite repetitive code as a developer would have had to do in the past, they are now able to refer to bits and pieces of their already-written code in order to reuse them.

Node.js is a server-side "JavaScript runtime environment that runs back-end application" (Aggarwal, 134). Developers are able to use Node.js to download several third party packages as well as push new updates to an already deployed website quickly and easily. In addition, Node.js allows for websites to function extremely quickly as it works on useful routines in the background rather than stalling as it waits for a user response. Ioannis K. Chaniotis, Kyriakos-Ioannis D. Kyriakou, and Nikolaos D. Tselikas actually tested out Node.js' processing speed and how it compares to other existing server-side frameworks in their paper "Is Node.js a viable option for building modern web applications? A performance evaluation study." In the study, Node.js was compared to other tools such as Apache and Nginix, and was proven to be significantly faster than both of them. "In the I/O test each server responds with a 'hello world' string. According to Fig. 8, which depicts the requests per second against concurrent connections, Nginx is more than 2.5 times faster in I/O operations than Apache, while Node.js outperforms both" (Chaniotis, 1035). Jing Huang and Lixiong Cai of the Zhejiang Science and Technology University of Hangzhou, China, also experimented with Node.js to evaluate its capabilities and found that it is also extremely fast when given large amounts of data. "Because of the single thread non-blocking model, Node.js can deal with big data and high concurrency in network communication rapidly and efficiently when facing big data and high frequency situations" (Huang, 6).

Express.js is a minimalist web framework for Node.js. Express is responsible for processing different user requests to a server, and enables developers to write efficient HTML code by allowing the inclusion of JavaScript statements within the HTML file. Express.js working in conjunction with Node.js results in an extremely fast functioning website. MongoDB is a database tool that stores data in key value pairs instead of in a grid of columns and rows (Aggarwal, 134). It essentially simplifies the job for the back-end developer as the data is now compiled in a similar format to objects in JavaScript. The simplification of MongoDB allows for developers to update and complete back end development much more quickly than before.

Essentially, stacks like the MERN stack enable developers to create beautiful, well-functioning websites much faster than before, as well as significantly increase the speed at which these websites run.

2.0.6 Other Frameworks/Libraries

A couple of other useful tools that assist developers in keeping up with the fast-paced standard of modern day web development include jQuery and Bootstrap.

Unlike the MERN stack, jQuery and Bootstrap are frameworks and libraries, rather than actual programming languages. They both essentially build and improve upon existing methodologies of web development. Specifically, jQuery simplifies the syntax and concepts of JavaScript, and Bootstrap contains a vast resource of already designed CSS elements. JQuery was developed in 2005 upon the idea that current JavaScript syntax could be improved upon. And in fact, the key difference that most people will notice when first using jQuery is the extremely simple syntax relative to that of JavaScript. In addition, jQuery also enables developers to manipulate HTML/CSS elements outside of the actual HTML/CSS file, this idea was revolutionary at the time of jQuery's inception. "Up to that point, I didn't realize that JavaScript code could be beautiful and elegant. Looking at Prototype inspired me to want to build something even better and add capabilities like manipulating HTML in the Document Object Model [DOM]" (Severance, 7). Nowadays, many developers still use jQuery in their projects because of its simplicity and also because it allows them to complete updates and development goals more quickly. Moreover, jQuery is an open source tool, meaning that it is constantly being updated and refined by developers from all around the world.

Bootstrap is a CSS framework that possesses a huge library of already-designed elements and is one of the most popular front-end development frameworks today. "The most popular ones today [CSS Libraries] are Bootstrap and Foundation..." (Temere, 1). In the past, developers used to have to individually style each and every element on a website with traditional CSS. This means in order to make a button a certain color, size, and shape, they would have had to manually code for each of these specifications. But with Bootstrap, they are now able to simply type in an existing button type and instantly produce a beautiful already-styled button that Bootstrap provides. The majority of websites use the Bootstrap library simply because of the convenience it provides when actually coding the website. However, the main feature that Bootstrap provides that makes it almost an essential to the modern day developer is its responsive capabilities. A glaring issue with a lot of websites today is the fact that they fail to resize according to the dimensions of the screen they are being viewed from. A website viewed on a desktop is not going to look as good on the much smaller screen of a smartphone. So with Bootstrap, developers are able to easily manipulate the elements on the webpage and have them respond according to the screen size they are being viewed from. "Bootstrap takes the one framework, every device approach. It can easily and efficiently scale websites and applications with a single code base. That means anything from phones to tablets and to desktops with CSS media queries" (Temere, 11).

2.0.7 Current Alternatives to My Website

It is without a doubt that students around the country rely on online rating/forum websites to develop an understanding of a specific professor at their institution before they consider enrolling in their class. Websites such as Ratemyprofessors.com, ProfessorPerformance.com, and PassCollege.com are all forum-styled platforms where users can come and give a specific professor at a college a review and a rating. Amongst these platforms, Ratemyprofessors.com is easily the most popular boasting a database of over 8000 schools, 1.7 million professors, and over 19 million ratings. The website also has an estimated 6 million visitors a month. Although Ratemyprofessors.com is extremely popular, it does not mean that its contents are valid or true. In fact, most professors on the website possess reviews and ratings from an extremely small percentage of their actual students. "Does ratemyprofessors.com really rate my professor?" by Otto et al. (2008) explores the validity of the data provided by Ratemyprofessors.com through statistical analysis. "However, it is also true that these ratings represent a small percentage of all students who register for courses taught by a professor. Many professors have only one rating, and even some of the most heavily rated professors, with 40 to 50 ratings, have had a great deal more students in their classes" (Otto, 356). This issue of professors receiving reviews only from a small group of students subjects the data provided by Ratemyprofessors.com to selection bias, due to the fact that only a small group of students are actually willing to go out of their way and give their professor a rating and write them a review. This small sample size inherently skews the data and ratings of the professors. The fact that Ratemyprofessors.com receives over 6 million monthly users and only possesses 19 million ratings, despite having been founded in 1999 is shocking. These statistics suggest that less than 2% of Ratemyprofessors.com's visitors actually leave a rating. ClassStats attempts to be more efficient and focus on converting visitors who search for ratings, to visitors who contribute ratings.

3 Methods

Once the preliminary research was compiled on the characteristics of a good website, the actual execution of the project was completed in four main phases.

Phase one consisted of the user research phase, validating initial assumptions and user demand. Phase two consisted of UI/UX design, actually materializing the envisioned product. Phase three consisted of database architecture, this is a database-heavy web application and so carefully organizing and structuring each part of the database is essential. And finally, phase four consisted of developing the website, putting everything together to create ClassStats.

3.0.1 Phase One: User Research

During this phase, the blueprint for what's to come was formulated. Research needed to be conducted in order to confirm the demand for the product. Thus, four main hypotheses needed to be supported.

H1: The majority of students use course review websites to aid them during the course registration process. (Figure 2)

H2: Ratemyprofessors.com is the most popular course review website. (Figure 3)

H3: The majority of students have never written a review. (Figure 4)

H4: The majority of students are unable to find what they are looking for when they visit these course review websites. (Figure 5)

The first hypothesis needed to be validated because the demand for a new and improved course review website is necessary to drive the motivation to create this product. There is no point in developing a website that no one is going to use. The second hypothesis assists in the establishment of a baseline for what ClassStats could become. If Ratemyprofessors.com proved to be the most popular website, then it would serve as the standard to wit ClassStats attempts to surpass. The third hypothesis is a personal assumption developed from the current statistics of Ratemyprofessors.com. The fact that the course review giant experiences roughly 6 million monthly visitors but contains significantly less course reviews, suggests that Ratemyprofessors.com does a poor job at converting visitors into contributors, meaning that the majority of students that visit Ratemyprofessors.com do not write reviews. And the potential confirmation of the fourth hypothesis further establishes a need for a new and improved course review alternative.

Throughout the Summer of 2019, a survey attempting to validate these hypotheses was administered to students at Occidental College as well as other select universities around the United States. The survey results validated hypotheses #1-3, however it seemed to have also invalidated hypothesis #4, almost 78% of students report that they are typically able to find what they are looking for when they visit Ratemyprofessors.com.

Although the initial results suggested that the majority of students are able to find what they are looking for, a separate finding from the survey revealed that this may actually not be the case. Figure 6 illustrates that although students think that reviews provided by

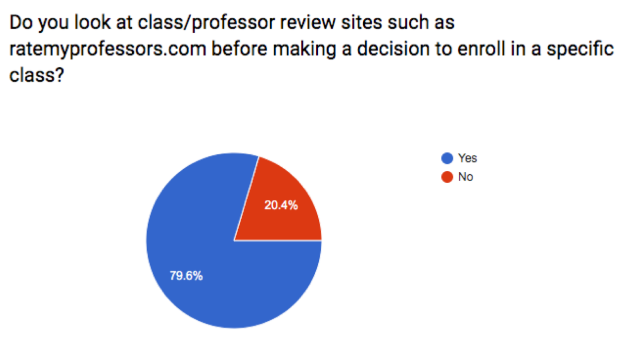


Figure 2: Survey Results H1

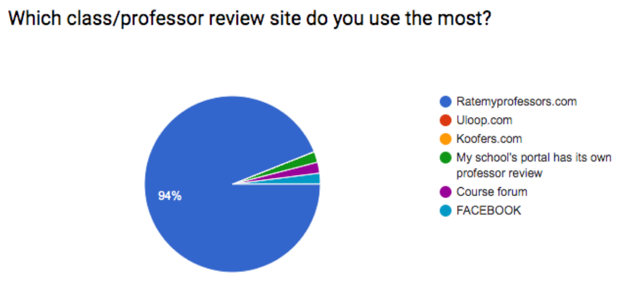


Figure 3: Survey Results H2

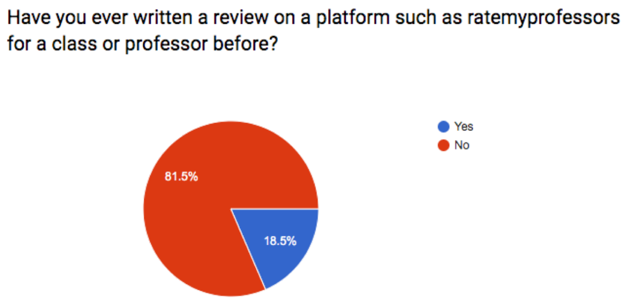


Figure 4: Survey Results H3

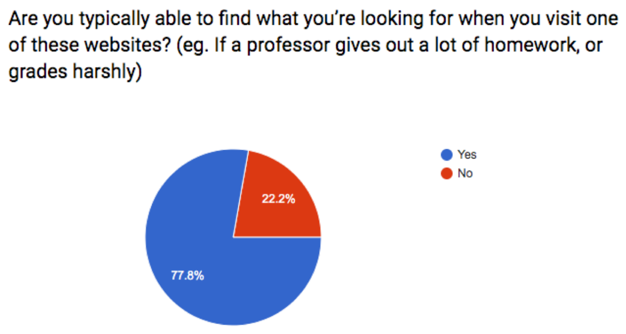


Figure 5: Survey Results H4

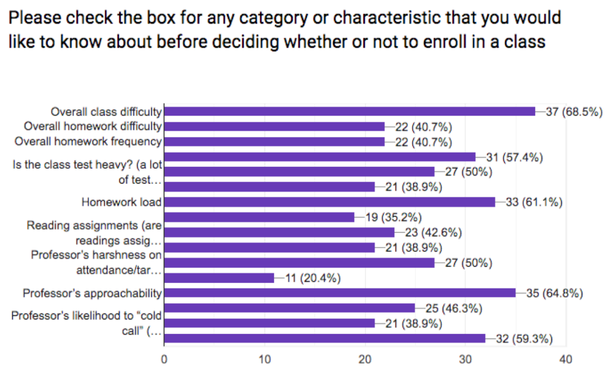


Figure 6: Survey Results

Ratemyprofessors.com are sufficient, they are clearly yearning for more specific pieces of information. Figure 6 illustrates the result of a portion of the survey that prompted students to select which of the specific course characteristics listed (as well as any they would like to submit) they would like to see on a course review website. Over 98% of students clicked on at least one of these specific characteristics, suggesting that there is an appeal for specificity in reviews for students. The results from this phase provided adequate reasoning to actually build ClassStats and even elicited further insight as to how it can be a better alternative to Ratemyprofessors.com.

3.0.2 Phase Two: UI/UX Design

During this phase, low-fidelity wireframes were created of how ClassStats would eventually look and feel. A mockup of the user-flow was designed that attempted to solve the various issues Ratemyprofessors.com retains. In order to specifically identify which of the flaws from Ratemyprofessors.com needed to be remedied, user interviews were conducted. Four user interviews were completed with students from Occidental College, University of Virginia, and Lehigh University. During these interviews, techniques such as active listening and open-ended questions were utilized in order to unbiasedly identify specific user pain points with current course review websites. One of the biggest takeaways from these interviews was the answer to why the majority of students do not submit reviews on Ratemyprofessors.com, because they never think to! "I guess I just never thought to [submit a review], because it seems time consuming..." (Elaine Cheng (Student at UVA), Interview by Julian Chan. Personal interview. Los Angeles, October 6, 2019.). Currently the user experience on Ratemyprofessors.com encourages users to view ratings, not contribute to them. In fact, a user needs to specifically go out of their way to submit a review on Ratemyprofessors.com, this process includes scrolling all the way to the bottom of the page and then filling out a form with over 10 inputs in order to submit one review. This process is extremely inefficient and evidently does a poor job at encouraging user reviews. It is because of this poor user experience that the majority of students don't even think to submit a review, and the ones that do submit reviews, do so only because they have extrinsic motivations. This actually skews the data because reviews are being completed by students who have personal reasons to write these reviews, this can result in overwhelmingly positive or negative ratings for professors. Thus, ClassStats was designed with this key goal in mind, convert visitors into contributors by intertwining the user's ability to view ratings, with the ability to provide ratings, into one cohesive user experience.

3.0.3 Phase Three: Database Architecture

In phase three, the database for ClassStats was designed. Due to the nature of this website a carefully thought-out database was essential to ensure the best possible user experience. Because ClassStats' success is solely reliant on the delivery of good quality data provided in the form of course ratings, a scalable, high-performing, reliable database language was needed, thus MySQL was chosen for this project. ClassStats' database is designed with three things in mind. The first is that each schema has a column to join with other tables, ensuring the possibility to join multiple different tables to provide multiple results. The second is that each table is minimal and only contained relevant attributes as to prevent unnecessary indexing thus improving runtime for queries. And the third is that each schema is designed as industry-standard as possible, meaning that each part of the database is carefully segmented ensuring the possession of relevant information, as opposed to placing all the data in one giant table. Over 16 schemas was created during this process and included dependent entities as well as identifying relationships. The database for ClassStats is as follows:

General Information
- Students: StudentID, Email, Password, Name, Major
- Instructors: InstructorID, Name, DeptID
- Departments: DepartmentID, Name
- Courses: CourseID, Name, InstructorID, DeptID
- Takes: StudentNum, CourseNum, Semester
- Teaches: CourseID, InstructorID, Semester

Ratings
- Class_Enjoyment: CourseID, Rating
- Class_Difficulty: CourseID, Easy, Medium, Hard
- Class_Type: CourseID, Lecture, Discussion
- Class_Usefulness: CourseID, Useful, NotUseful
- Attendance_Attn: CourseID, Attentive, Inattentive
- Exam_Difficulty: CourseID, Easy, Medium, Hard
- Homework_Load: CourseID, Heavy, NotHeavy
- Prof_Approachability: CourseID, Approachable, NotApproachable
- Prof_Rating: CourseID, Rating
- Test_Heavy: CourseID, Heavy, NotHeavy

Note that each of these ratings were chosen based on a majority vote from the user survey. Each characteristic was selected by more than 50% of students when prompted on which course characteristic they would find useful in aiding in their decision to enroll in a class (Refer to figure 6). The ratings work by incrementing a specific attribute within the specified rating. For example, if a user found a class was extremely useful and therefore clicks on the "Useful" option for Class_Usefulness, the database would increment the "Useful" column by one, thus recording the submission. Each rating also possesses a "count" attribute that keeps track of how many students provided that specific rating for that specific course in order to provide an average rating for each characteristic. So if "Useful" contains a value of 8 and "NotUseful" contains a value of 2, "Count" would contain a value of 10, and the displayed ratings are Useful = 80%, and NotUseful = 20%.

3.0.4 Phase Four: Development

During this phase ClassStats finally became a reality. The stack used for this web application was Node.js for the server-side language, MySQL for the database, and React.js, HTML, and CSS for the front-end. In addition, libraries such as Bootstrap were used in order to incorporate responsive capabilities into the website. In order to complete this project within the timeframe, a lot of preparation and practice had to take place with specific web development frameworks. Specifically, many practice websites/web apps utilizing Node.js and Bootstrap were created in order to prepare for this project. In addition, React.js was also learned concurrently during each of the four phases, as to ensure quick and seamless implementation once development begun. Node.js was selected for the server-side language for this project due to its ability to work extremely quickly. With Node.js, updates and bug fixes are able to be pushed to a live site within minutes and require minimal unnecessary work. Node.js also has a vast library of dependencies enabling efficient integration with front-end languages and files. Some of these advantages include, quick integration with MySQL database, structured organization of HTTP requests, and extremely powerful debugging capabilities. MySQL was selected as the database language for this project due to its scalability and reliability. MySQL is famous for its ability to handle large quantities of data. And so due to ClassStats' requirement to accumulate large amounts of data in the form of user ratings, MySQL is the clear choice for this project as opposed to a non-relational database such as MongoDB. React.js was probably the most important framework for this project, not because of it's extremely fast processing time as discussed in the section 2.0.5, but because of its ability to dynamically load data from the server without page refresh. As discussed above, one of the biggest differences between ClassStats and Ratemyprofessors.com is the newly combined user experience of both viewing ratings, and providing ratings. ClassStats aims to encourage visitors to submit ratings in a way that does not impair their user experience. And so utilizing a framework such as React.js that enables users to submit ratings without constant page refresh is imperative. Moreover, React.js allows clients to render elements extremely quickly, allowing for real time updates on the client side. Once each of these frameworks and languages were selected, the development process ensued. After several weeks of coding, simultaneously developing both the front-end and the back-end, as well as integration with the database, ClassStats was finally created.

3.0.5 Evaluation

This project should be evaluated on its ability to perform to the specifications listed in the original proposal and initial vision. Specifically, it should be evaluated based on if it was successfully created utilizing the specified tools such as React.js, MySQL, and Node.js, and if the functionality of the website meets those specified in the scope of the proposal.

4 Discussion: Results

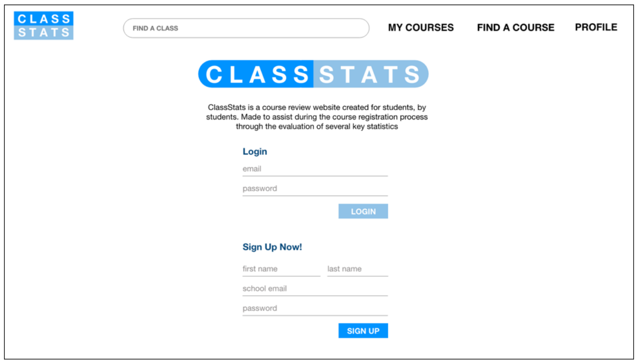
4.0.1 Assumptions

Due to the time limitations of this project, some assumptions had to be made in order to create the best possible product. Specifically, the solution to one of the initial problems regarding the validity of course reviews was created under the premise that ClassStats would partner with Occidental College. The assumption is that ClassStats would have an official partnership with Occidental, granting ClassStats access to the student database providing information such as student emails, student IDs, and course history. This way, ClassStats would have the necessary tools to ensure that students can only submit course reviews and ratings to classes they have actually taken or are currently enrolled in. This assumption is not too misguided and is actually inspired by a similar system the University of Virginia currently endorses. The University of Virginia has a course review website specific to their university and their curriculum, this website enables only their students to view the site contents and provides information such as student course history (Elaine Cheng (Student at UVA), Interview by Julian Chan. Personal interview. Los Angeles, October 6, 2019.). With this assumption, ClassStats is able to present the visiting user their currently enrolled courses, along with specifically tailored pop-up messages based on the student's course history. It is this underlying assumption that enables ClassStats to surpass Ratemyprofessors.com's current user experience, achieving the goal of converting visitors into contributors.

4.0.2 The Final Product - User Flow

The final product for ClassStats contains dummy student/course history data, but contains real course, instructor, and department information pulled from the Occidental course list website counts.oxy.edu. Data pulled from this website includes, course ID, course name, semester, and instructor name. The website is fully functional and responsive to multiple devices, below displays each wireframe with its mobile counterpart along with an explanation.

1. The initial screen for ClassStats is a login/sign up page. Users must sign up with their Occidental college email in order to access the website's contents. Internally, the ClassStats server checks to see if the school email is located in the Students database ensuring the user's enrollment in the college. Once the email has been verified, the user successfully signs up and a hashed password is stored in the database to that specific email. Once the user logs in, the client (device ClassStats is being viewed from) stores a cookie with the user and session's information enabling the user to exit and re-enter the website without having to re-login. The session times out after 2 hours and if the user has not been active on the website, ClassStats will prompt the user to login again. An additional feature throughout the website is a middleware function. This middleware repeatedly checks to see if the client contains a cookie possessing the user's information and if the session is still active, if both return true then the server will render the webpage, if not, the server will redirect the user to the login/sign up page. This middleware function is extremely useful as it prevents non-Occidental College students from accessing the website through copying and pasting a link. (Refer to figure 7 and figure 8).


CLASS STATS

FIND A CLASS

MY COURSES FIND A COURSE PROFILE

CLASS STATS

ClassStats is a course review website created for students, by students. Made to assist during the course registration process through the evaluation of several key statistics

Login

email

password

LOGIN

Sign Up Now!

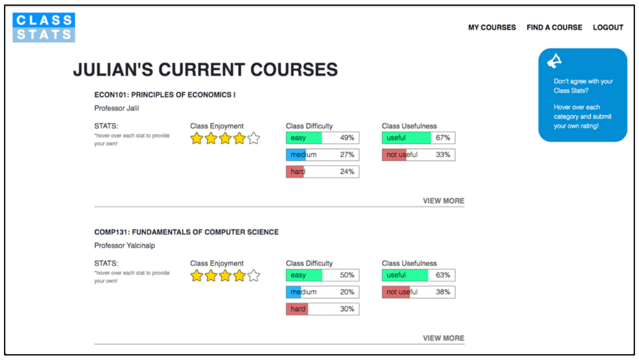
first name last name

school email

password

SIGN UP

Figure 7: ClassStats Login/Signup (Desktop)


CLASS STATS

MY COURSES FIND A COURSE LOGOUT

JULIAN'S CURRENT COURSES

Don't agree with your Class Stats? Hover over each category and submit your own rating!

ECON101: PRINCIPLES OF ECONOMICS I

Professor Jalil

STATS: *hover over each stat to provide your own!

Class Enjoyment	Class Difficulty		Class Usefulness	
★★★★☆	easy	49%	useful	67%
	medium	27%	not useful	33%
	hard	24%		

VIEW MORE

COMP131: FUNDAMENTALS OF COMPUTER SCIENCE

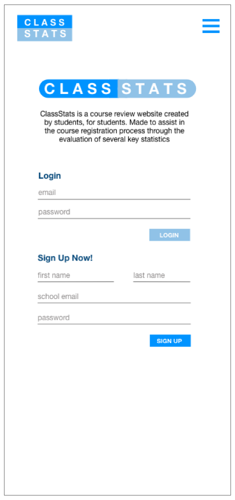
Professor Yalcinalp

STATS: *hover over each stat to provide your own!

Class Enjoyment	Class Difficulty		Class Usefulness	
★★★★☆	easy	50%	useful	63%
	medium	20%	not useful	38%
	hard	30%		

VIEW MORE

Figure 9: ClassStats Homepage (Desktop)


CLASS STATS

CLASS STATS

ClassStats is a course review website created by students, for students. Made to assist in the course registration process through the evaluation of several key statistics

Login

email

password

LOGIN

Sign Up Now!

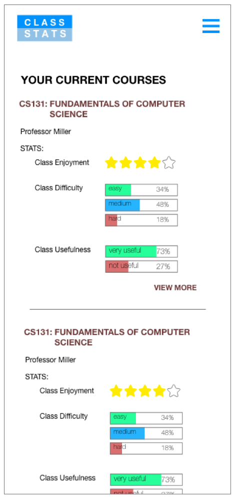
first name last name

school email

password

SIGN UP

Figure 8: ClassStats Login/Signup (Mobile)


CLASS STATS

YOUR CURRENT COURSES

CS131: FUNDAMENTALS OF COMPUTER SCIENCE

Professor Miller

STATS:

Class Enjoyment	★★★★☆	
Class Difficulty	easy	34%
	medium	48%
	hard	18%
Class Usefulness	very useful	73%
	not useful	27%

VIEW MORE

CS131: FUNDAMENTALS OF COMPUTER SCIENCE

Professor Miller

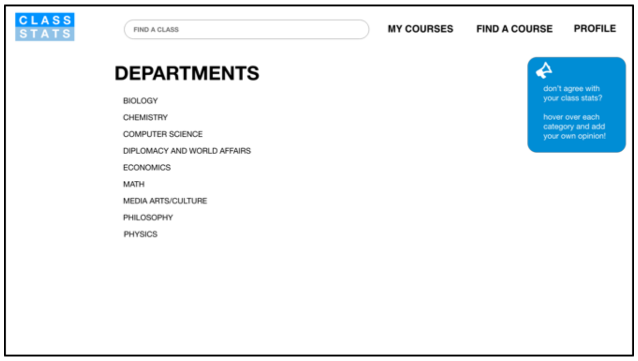
STATS:

Class Enjoyment	★★★★☆	
Class Difficulty	easy	34%
	medium	48%
	hard	18%
Class Usefulness	very useful	73%

Figure 10: ClassStats Homepage (Mobile)

2. Once the user logs in they are presented with the home page that lists their current courses along with three ratings for each course. The ratings displayed are an aggregate average of all the ratings submitted for that specific course taught by that specific professor. The goal with this home screen is to encourage the user to provide ratings for the course they are currently taking. The theory is that when a user logs in to ClassStats, presumably in search for ratings for a potential class in the upcoming semester, they will view the ratings for their currently enrolled courses and provide their own rating. For example, if a user logs in and sees that only 67% of students found Econ 101: Principles of Economics useful, they can hover over the "useful" bar and click to submit their own rating if they also found it useful. Once they do, the database is updated immediately and the updated percentages are displayed. (Refer to figure 9 and figure 10).

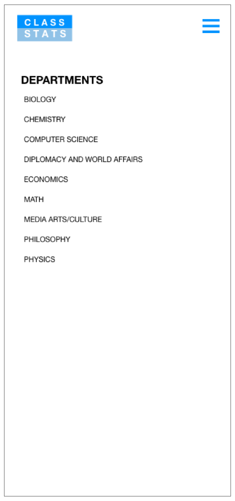
3. If the user clicks on "Find a Course" in the navigation bar they are presented with the Departments page. This page pulls all the departments from the database with a GET request and presents them in the form of anchor tags (¡a¿ ¡/a¿) in HTML for the client. The query used for this page is "SELECT * FROM Departments." (Refer to figure 11 and figure 12).


CLASS STATS
FIND A CLASS
MY COURSES
FIND A COURSE
PROFILE
don't agree with your class stats?
hover over each category and add your own opinion!

DEPARTMENTS

BIOLOGY
CHEMISTRY
COMPUTER SCIENCE
DIPLOMACY AND WORLD AFFAIRS
ECONOMICS
MATH
MEDIA ARTS/CULTURE
PHILOSOPHY
PHYSICS

Figure 11: ClassStats Departments (Desktop)

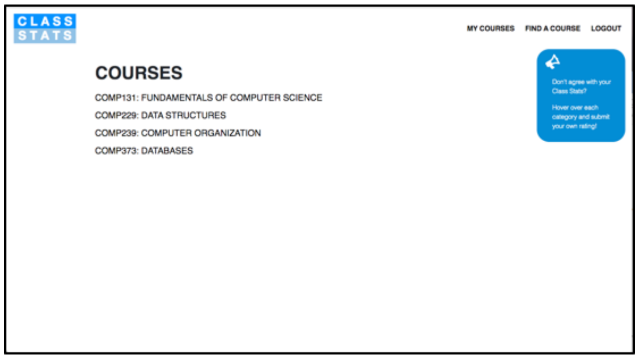


DEPARTMENTS

BIOLOGY
CHEMISTRY
COMPUTER SCIENCE
DIPLOMACY AND WORLD AFFAIRS
ECONOMICS
MATH
MEDIA ARTS/CULTURE
PHILOSOPHY
PHYSICS

Figure 12: ClassStats Departments (Mobile)

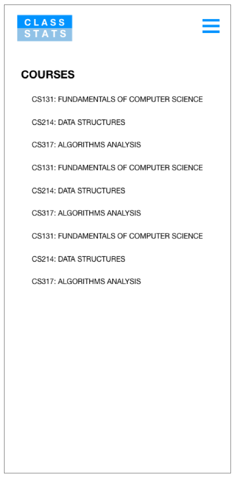
4. Once the user clicks on a specific department, they are presented with all the courses listed by that department. Data is pulled from the server with a GET request similar to Departments. (Refer to figure 13 and figure 14).


CLASS STATS
MY COURSES
FIND A COURSE
LOGOUT
Don't agree with your Class Stats?
Hover over each category and submit your own rating!

COURSES

COMP131: FUNDAMENTALS OF COMPUTER SCIENCE
COMP229: DATA STRUCTURES
COMP239: COMPUTER ORGANIZATION
COMP373: DATABASES

Figure 13: ClassStats Courses (Desktop)

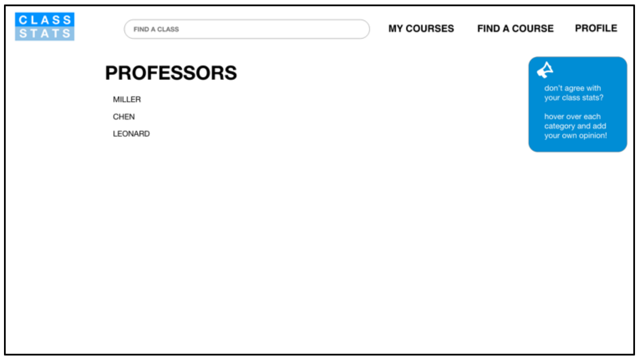


COURSES

CS131: FUNDAMENTALS OF COMPUTER SCIENCE
CS214: DATA STRUCTURES
CS317: ALGORITHMS ANALYSIS
CS131: FUNDAMENTALS OF COMPUTER SCIENCE
CS214: DATA STRUCTURES
CS317: ALGORITHMS ANALYSIS
CS131: FUNDAMENTALS OF COMPUTER SCIENCE
CS214: DATA STRUCTURES
CS317: ALGORITHMS ANALYSIS

Figure 14: ClassStats Courses (Mobile)

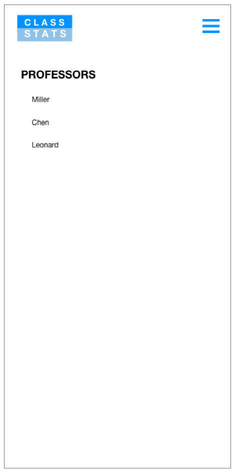
5. Once the user clicks on a specific class, they are presented with all the professors that teach that specific course. Data is pulled from the server with a GET request similar to Departments. (Refer to figure 15 and figure 16).


CLASS STATS
FIND A CLASS
MY COURSES
FIND A COURSE
PROFILE
don't agree with your class stats?
hover over each category and add your own opinion!

PROFESSORS

MILLER
CHEN
LEONARD

Figure 15: ClassStats Professors (Desktop)


CLASS STATS

PROFESSORS

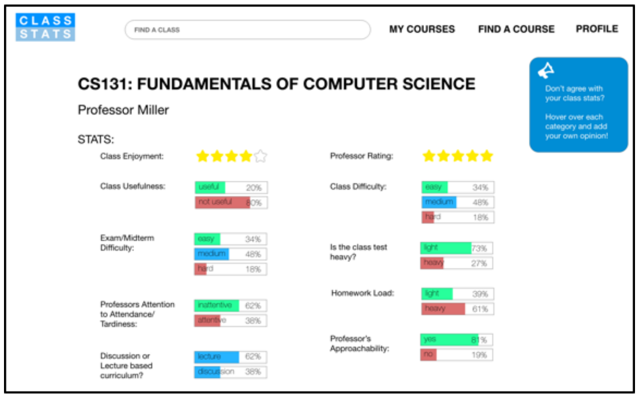
Miller

Chen

Leonard

Figure 16: ClassStats Professors (Mobile)

6. Once the user finally accesses a specific class taught by a specific professor, they are presented with the “Class Stats” for the course. Each “Class Stat” was chosen based on the results from the user survey in phase one. Every rating was defined by at least 50% of users as important in assisting in their decisions during course registration. The ratings are presented in two forms, a rating out of five stars, and a percentage. All of the ratings are averaged based on the total number of submissions and present the real time data from the ClassStats database. The goal for this page is to allow the user to scan the statistics enabling them to make a more precise, informed decision on whether or not the class interests them or not. Some of these statistics are very essential during the decision making process. For example, “Class Usefulness” reflects the perception of previous students on the overall practicality of the course beyond the scope of academics. And “Discussion or Lecture based curriculum” is essential in assisting students in making an informed decision due to the fact that some students may find one teaching style more preferable over the other. (Refer to figure 17 and figure 18).


CLASS STATS

FIND A CLASS

MY COURSES FIND A COURSE PROFILE

CS131: FUNDAMENTALS OF COMPUTER SCIENCE

Professor Miller

STATS:

Class Enjoyment: ★★★★☆

Class Usefulness: useful 20%; not useful 80%

Exam/Midterm Difficulty: easy 34%; medium 48%; hard 18%

Professors Attention to Attendance/ Tardiness: inattentive 62%; attentive 38%

Discussion or Lecture based curriculum? lecture 62%; discussion 38%

Professor Rating: ★★★★★

Class Difficulty: easy 34%; medium 48%; hard 18%

Is the class test heavy? light 73%; heavy 27%

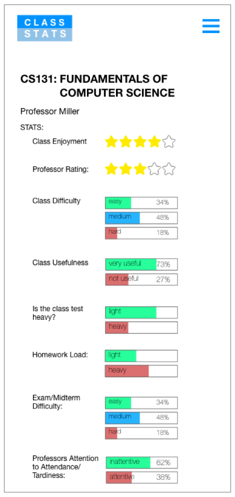
Homework Load: light 39%; heavy 61%

Professor's Approachability: yes 81%; no 19%

Don't agree with your class stats?

Hover over each category and add your own opinion!

Figure 17: ClassStats Ratings (Desktop)


CLASS STATS

CS131: FUNDAMENTALS OF COMPUTER SCIENCE

Professor Miller

STATS:

Class Enjoyment ★★★★☆

Professor Rating: ★★★☆☆

Class Difficulty: easy 34%; medium 48%; hard 18%

Class Usefulness: very useful 73%; not useful 27%

Is the class test heavy? light; heavy

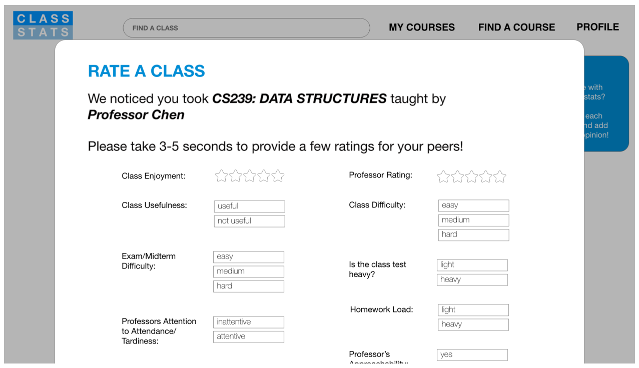
Homework Load: light; heavy

Exam/Midterm Difficulty: easy 34%; medium 48%; hard 18%

Professors Attention to Attendance/ Tardiness: inattentive 62%; attentive 38%

Figure 18: ClassStats Ratings (Mobile)

7. During the session, the server will search for a course the user had taken in the past and had also not provided a rating for, then it will prompt them to provide ratings for that course with a popup. This popup will only appear once during the session, as repeated popups would negatively affect the overall UX of the website. The user must submit at least 3 ratings to exit out of the popup, the goal here is for the user to realize how easy it is to submit ratings on ClassStats and to encourage this behavior moving forward. (Refer to figure 19 and figure 20).



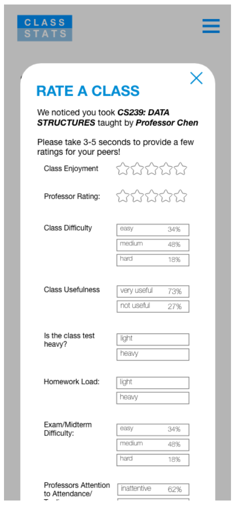
RATE A CLASS

We noticed you took ***CS239: DATA STRUCTURES*** taught by ***Professor Chen***

Please take 3-5 seconds to provide a few ratings for your peers!

Class Enjoyment:	☆☆☆☆☆	Professor Rating:	☆☆☆☆☆
Class Usefulness:	useful not useful	Class Difficulty:	easy medium hard
Exam/Midterm Difficulty:	easy medium hard	Is the class test heavy?	light heavy
		Homework Load:	light heavy
Professors Attention to Attendance/ Tardiness:	inattentive attentive	Professor's	yes

Figure 19: ClassStats Popup (Desktop)



RATE A CLASS

We noticed you took ***CS239: DATA STRUCTURES*** taught by ***Professor Chen***

Please take 3-5 seconds to provide a few ratings for your peers!

Class Enjoyment	☆☆☆☆☆
Professor Rating:	☆☆☆☆☆
Class Difficulty	easy 34% medium 48% hard 18%
Class Usefulness	very useful 73% not useful 27%
Is the class test heavy?	light heavy
Homework Load:	light heavy
Exam/Midterm Difficulty:	easy 34% medium 48% hard 18%
Professors Attention to Attendance/	inattentive 62%

Figure 20: ClassStats Popup (Mobile)

4.0.3 Concepts Used

There were many web development concepts used for this project. All of which combined to deliver a fully-functional, secure, REST API server. One of the main concepts used for this project was the utilization of HTTP requests. Also known as Hypertext Transfer Protocol, HTTP is used to structure requests and responses over the internet. HTTP requires data to be transferred from one point to another over a network. Some HTTP requests used for ClassStats include GET requests to render data on the client from the server, and POST requests to send user data from the client to the server in order to render specific elements or to update the database. An example of a GET request would be clicking on the "Computer Science" department, the server pulls all the courses listed under "Computer Science" and renders the data back onto the client. An example of a POST request would be during login/sign up, once a user signs up, the client uses a POST request so that the server can update the database with the newly defined password.

Another concept used for this project was the use of query string parameters. These parameters are key-value pairs that appear inside a URL and follow a question mark. These key-value pairs enable the client to send data to the server quickly and seamlessly through the URL of an HTTP request. An example of how a query string parameter was used for ClassStats is during the rendering of the ratings for a specific course. When the user has clicked onto a specific course to view its ratings, the URL contains query string parameters that notifies the server which data to pull from which table in the database. The URL could look something like "http://www.classstats.com/course?courseid=1615." In this URL, the query string parameter follows the '?' and is listed as "course ID = 1615." This parameter is then sent to the server notifying it to pull ratings from the database for the course where course ID = 1615. This then allows the server to render unique data to the client.

A third concept used for this project was the use of JSON files to send data to and from the server. Also known as JavaScript object notation, JSON files store simple data structures and objects that enable the ease of communication between server and client. An example of how JSON files is used in ClassStats is on the home page to render the specific name of the user that is logged in. Once the user logs in, the server sends a JSON file to the client that contains the user's information in the form of an object. Then all the client has to do to render the data from the server is call upon the user object as well as a key such as "name." So to render the specific name of the user on the home page, the client calls "user.name" to render that information.

A fourth concept use for this project was MySQL prepared statements. This concept is essential to ensure ClassStats is secure by preventing SQL injection. A form of hacking, SQL injection refers to the ability of the client to bypass the specified SQL query on the server by inputting a ';' or closing apostrophe followed by a new, destructive SQL query. In other words, a simple query that displays data for a specific user, could end up having a second query that deletes the entire database, if not handled correctly. The use of prepared statements prevents this from happening by specifying a set SQL query and expected SQL parameter. So if this parameter contains suspicious characters such as a semicolon or another SQL query, the server will return an error and prevent the destructive SQL query from ever executing. Another privacy protection protocol utilized for ClassStats is the use of hashed passwords. Hashing refers to the one-way transformation of a password, which then turns the password into another string. Once a password is hashed, it is virtually impossible to revert it back to the original string, as the new hashed string possesses different characters, takes on a different length, and is virtually indistinguishable from the original. For example, a password such as "Tigers123" could be hashed into a string such as "!3.ws#!m%$@dW." Thus, this prevents users from getting their passwords divulged if the database were ever breached.

4.0.4 Features Completed

The final product contains many notable features that make it both secure and efficient. ClassStats possesses full user authentication capabilities, meaning login and sign up are fully functional. A user can only signup on ClassStats if they have an official Occidental College email. Once they signup, their password is put through a hash function before being stored in the database. After the user has successfully logged in or signed up, ClassStats stores a cookie on the user's client containing the current session information as well as their user information. This cookie is then called upon before every web page is rendered through a middleware function which checks to see if the user is logged in before rendering a web page. This middleware function ensures the security of ClassStats and prevents non-Occidental College students from viewing course ratings. If the middleware finds that the user has not logged in, then the server will redirect the client back to the login/sign up page. In addition, because the user and session information is being stored on a cookie on the user's device, the user has the ability to exit out of ClassStats and reload it without having to login again.

In addition, ClassStats is fully dynamic, meaning that there is minimal hard code written for the front-end. ClassStats is consistently making queries to the database in order to render specific contents on the client. Some examples include the courses page that lists out all the offered courses by a specific department. When the page is accessed, ClassStats executes the MySQL prepared statement:

```
SELECT Courses.Name, Departments.Name AS
    Department
FROM Courses, Departments
WHERE Courses.DeptID = (
    SELECT DepartmentID
    FROM Departments
    WHERE Departments.name = ? )
AND Departments.Name = ? ;
```

Where all the '?' are replaced with the selected department name.

ClassStats also utilizes React.js to send data to the server through query string parameters. Once a user clicks on a specific course, React.js stores the user's course history in "State," a React.js concept that stores property values in the form of an object that belongs to a React.js component. And if the user submits a rating, React.js first checks to see if the user had actually taken that specific course by looping through the user's course history in "State." If the user's previous enrollment in the course is verified, React.js will send the data to the server to update the database, and the new aggregated rating average is displayed for the user without page refresh. This would not be possible without React.js as typical HTML form submissions require constant page refresh after each communication with the server. If ClassStats ratings were developed with regular HTML, then each time a user clicks on a specific rating, the page would have to refresh, which would negatively affect the overall UX of ClassStats.

Moreover, ClassStats contains an API that automatically searches the database for a specific course the user had previously enrolled in that they had not yet submitted ratings for, and auto-prompts them to submit ratings for it through a one-time popup. This popup forces users to submit course ratings in order to continue to view ClassStats' contents. Once the user submits their ratings, ClassStats stores a cookie on the client that indicates they have already received the popup, preventing ClassStats from prompting the user again.

5 Discussion: Ethical Ramifications & Considerations

An ethical consideration for ClassStats would be the misuse and exploitation of the service by students who provide unjust ratings due to personal feelings or opinions about a professor that have nothing to do with academia. These ratings will be hard to identify, however, there are potential solutions. For example, moving forward ClassStats will keep track of every student's rating submission, with that record, ClassStats could notify the admin that a specific student has been only submitting negative ratings for their classes. This notification would then spark further investigation and if it is found that the student is failing to provide quality ratings, they will be prohibited from submitting ratings all together. Another ethical consideration would be to include more diverse, unique course metrics. Currently, ClassStats displays the same course characteristics for every class. However, since no two classes are identical, the inclusion of unique characteristics would be useful in assisting students during course registration. For example, courses under the Math department could include unique course ratings specific to Mathematics that would be inappropriate to include for a History course.

6 Discussion: Strengths, Weaknesses, Future Work

One of the biggest strengths for this project was time management. The deadlines specified in the original proposal were consistently met and never needed any alterations. In addition, the comprehension and implementation of React.js was a success, however, to a slightly less degree than originally anticipated. Initially, React.js was to be implemented with the entire front-end for ClassStats. But as the weeks wore on, it didn't seem necessary to implement React.js for every part of the website, and actually would have been counter-productive. Ultimately, React.js was utilized for the sole purpose of displaying and submitting ratings, due to its dynamic capabilities of rendering new data from the server without page refresh.

A weakness for this project was the inability to deploy the website online before the specified Comps deadline. Currently an issue with ClassStats is its inability to perform under free server hosting plans. After trying to deploy ClassStats on a free server hosting website (heroku) while utilizing a free database host (clearDB), there were a few issues regarding max database connections. This is mainly due to the fact that ClassStats contains many SQL connections to render data for the client, and even though ClassStats performs perfectly locally, it fails to deliver under restrictive online database plans. A solution to this would be to minimize the amount of SQL connections ClassStats maintains and combine multiple SQL queries into one large query. However, this would negatively affect performance in regards to runtime so perhaps an alternative solution would be to just upgrade to a less restrictive database plan with a larger amount of max database connections per user.

Next steps for ClassStats includes potentially officially partnering with Occidental College. If ClassStats were able to partner with Occidental, it would possess the entire student database along with every student's course history, granting it the ability to finally provide robust, quality reviews for courses. In addition, ClassStats could potentially replace the existing course evaluation process Occidental currently enforces. Instead of having students submit a google form at the end of the semester to provide professor feedback, they could simply fill out their class' ClassStats ratings. It is without a doubt that course review websites are a necessary evil, even though Ratemyprofessors.com perpetuates biased, unjust reviews by students that have not even been verified, students will consistently rely on these services for insight during course registration. So if students are going to continue to utilize course review websites, why not provide them with a better, more reliable alternative? The goal moving forward is to revisit the user experience design of ClassStats and to test each feature with multiple rounds of A/B testing and QA testing, this would ensure that ClassStats exhibits the most optimal user experience for converting visitors into contributors. Moreover, a ClassStats mobile app would also be ideal as smartphone users have consistently resorted to utilizing mobile apps as opposed to their website counterparts. Having a ClassStats mobile app would open up many doors to opportunities that web applications are simply unable to provide. For instance, having a mobile app would enable the use of push notifications. So during the semester, the ClassStats app could hypothetically notify a student to submit a course rating, this notification would only require them to provide one, single rating. This use of notifications would theoretically increase the amount of course ratings and expand upon the user experience of ClassStats beyond the website. With a re-organized UX, addition of a mobile app, and the potential partnership with Occidental, there is no reason to believe that ClassStats will not be replacing Ratemyprofessors.com as the go-to course review website for students at Occidental College.

References

[1] Aggarwal, Sanchit. "Modern Web-Development Using ReactJS." International Journal of Recent Research Aspects, vol. 5, no. 1, Mar. 2018, pp. 133–137.

[2] Aggarwal, Sanchit, and Jyoti Verma. "Comparative Analysis of MEAN Stack and MERN Stack." International Journal of Recent Research Aspects, vol. 5, no. 1, Mar. 2018, pp. 133–137.

[3] Almeida, Fernando, and José Monteiro. "The Role of Responsive Design in Web Development." Webology, vol. 14, no. 2, Dec. 2017, pp. 1–18., www.webology.org/2017/v14n2/a157.pdf.

[4] Baker, Stewart C. "Making It Work for Everyone: HTML5 and CSS Level 3 for Responsive, Acces-

sible Design on Your Librarys Web Site." Journal of Library and Information Services in Distance Learning, vol. 8, no. 3-4, 2014, pp. 118–136., doi:10.1080/1533290x.2014.945825.

[5] Blansit, B. Douglas.*An Introduction to Cascading Style Sheets (CSS)*, Journal of Electronic Resources in Medical Libraries, vol. 5, no. 4, 2008, pp. 395–409., doi:10.1080/15424060802453811.

[6] Chaniotis, Ioannis K., et al. "Is Node.js a Viable Option for Building Modern Web Applications? A Performance Evaluation Study." Computing, vol. 97, no. 10, 2014, pp. 1023–1044., doi:10.1007/s00607-014-0394-9.

[7] Comeaux, David J. "Web Design Trends in Academic Libraries—A Longitudinal Study." Journal of Web Librarianship, vol. 11, no. 1, 2016, pp. 1–15., doi:10.1080/19322909.2016.1230031.

[8] Cyr, Dianne and Ilsever, Joe and Bonanni, Carole and Bowes, John. (2004). Website Design and Culture: An Empirical Investigation. 33-.

[9] Ducasse, Stéphane, et al. "Seaside: A Flexible Environment for Building Dynamic Web Applications." IEEE Software, vol. 24, no. 5, 2007, pp. 56–63., doi:10.1109/ms.2007.144.

[10] Ducasse, Stéphane, et al. "Seaside: A Flexible Environment for Building Dynamic Web Applications." IEEE Software, vol. 24, no. 5, 2007, pp. 56–63., doi:10.1109/ms.2007.144.

[11] Elrom, Elad. "CSS, Bootstrap, and Responsive Design." Pro MEAN Stack Development, 2016, pp. 131–164.

[12] Glassman, Nancy R., and Phil Shen. "One Site Fits All: Responsive Web Design." Journal of Electronic Resources in Medical Libraries, vol. 11, no. 2, 2014, pp. 78–90., doi:10.1080/15424065.2014.908347.

[13] Hsieh, Tsuei-Ju Tracy. "Multiple Roles of Color Information in the Perception of Icon-Type Images." Color Research and Application, vol. 42, no. 6, 2017, pp. 740–752., doi:10.1002/col.22140.

[14] Huang, Jing, and Lixiong Cai. "Research on TCP/IP Network Communication Based on Node.js." 2018, doi:10.1063/1.5033779.

[15] Jokinen, Jussi P P, et al. "Relating Experience Goals With Visual User Interface Design." Interacting with Computers, vol. 30, no. 5, 2018, pp. 378–395., doi:10.1093/iwc/iwy016.

[16] Lin, Chiuhsiang Joe, and Lai-Yu Cheng. "Product Attributes and User Experience Design: How to Convey Product Information through User-Centered Service." Journal of Intelligent Manufacturing, vol. 28, no. 7, 2015, pp. 1743–1754., doi:10.1007/s10845-015-1095-8.

[17] "Responsive Web Design Tenets." Building Responsive Data Visualization for the Web, 2015, pp. 24–69., doi:10.1002/9781119209560.ch2.

[18] Romano, Nicholas C., et al. "Architecture, Design, and Development of an HTML/JavaScript Web-Based Group Support System." Journal of the American Society for Information Science, vol. 49, no. 7, 1998, pp. 649–667., doi:10.1002/(sici)1097-4571(19980515)49:7¡649::aid-asi6¿3.0.co;2-1.

[19] Martin, Erik J. "Is Agile Web Development All It's Cracked up to Be?" EContent, vol. 41, no. 2, Spring 2018, pp. 26–30.

[20] Miller-Francisco, Emily. "Creating Dynamic Websites Using JQuery." Computers in Libraries, vol. 30, no. 6, July 2010, pp. 26–28.

[21] Otto, James, et al. "Does Ratemyprofessor.com Really Rate My Professor?" Assessment and Evaluation in Higher Education, vol. 33, ser. 4, 23 July 2008, pp. 355–368. 4, doi:10.1080/02602930701293405.

[22] Schneider, Christoph, et al. "Digital Nudging." Communications of the ACM, vol. 61, no. 7, 2018, pp. 67–73., doi:10.1145/3213765.

[23] Severance, Charles. "John Resig: Building JQuery." Computer, vol. 48, no. 5, 2015, pp. 7–8., doi:10.1109/mc.2015.135.

[24] Subramanian, Vasan. "Pro MERN Stack." 2017, doi:10.1007/978-1-4842-2653-7.

[25] Temere, Befekadu Mezgebu. "Responsive Web Application Using Bootstrap and Foundation." Helsinki Metropolia University of Applied Sciences, 2017.

[26] Theopilus, Yansen, et al. "Persuasive-Universal Design Model for Creating User Experience in Product to Solve Behavior Problems." 2018, pp. 2–8., doi:10.1063/1.5042929.

[27] Tzafilkou, Katerina, and Nicolaos Protogeros. "Diagnosing User Perception and Acceptance Using Eye Tracking in Web-Based

End-User Development.” Computers in Human Behavior, vol. 72, July 2017, pp. 23–37., doi:10.1016/j.chb.2017.02.035.

[28] Wilkinson, Jaci. “Accessible, Dynamic Web Content Using Instagram.” Information Technology and Libraries, vol. 37, no. 1, 2018, p. 19., doi:10.6017/ital.v37i1.10230.

[29] Williams, Sam, and Sam Williams. “The Airbnb Story – Daily Book Notes – Medium.” Medium.com, Medium, 31 Jan. 2018, medium.com/daily-book-notes/the-airbnb-story-d6267a09c3c1.

[30] Zakas, Nicholas C. “The Evolution of Web Development for Mobile Devices.” Communications of the ACM, vol. 56, no. 4, 2013, p. 42., doi:10.1145/2436256.2436269.

[31] Zheng, Pai, et al. “User-Experience Based Product Development for Mass Personalization: A Case Study.” Procedia CIRP, vol. 63, 2017, pp. 2–7., doi:10.1016/j.procir.2017.03.122.