

6-11-2018

SCU Evals

Fredrik Blomqvist

Santa Clara University, fblomqvist@scu.edu

Joseph Théberge

Santa Clara University, jtheberge@scu.edu

Follow this and additional works at: https://scholarcommons.scu.edu/cseng_senior



Part of the [Computer Engineering Commons](#)

Recommended Citation

Blomqvist, Fredrik and Théberge, Joseph, "SCU Evals" (2018). *Computer Engineering Senior Theses*. 118.

https://scholarcommons.scu.edu/cseng_senior/118

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Computer Engineering Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

Santa Clara University
DEPARTMENT of COMPUTER ENGINEERING

Date: June 11, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY
SUPERVISION BY

Fredrik Blomqvist and Joseph Théberge

ENTITLED

SCU Evals

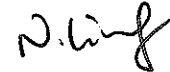
BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING



THESIS ADVISOR



DEPARTMENT CHAIR

SCU EVALS

by

Fredrik Blomqvist and Joseph Théberge

SENIOR DESIGN PROJECT REPORT

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California

June 11, 2018

SCU Evals

Fredrik Blomqvist
Joseph Théberge

Department of Computer Engineering
Santa Clara University
June 14, 2018

ABSTRACT

Evaluating professors and classes is a common task at universities across the United States. It helps professors improve courses, and guides students when choosing classes tailored to their learning style. The current available evaluation platforms do not offer intuitive, up-to-date, and encouraging solutions to this problem; thus, SCU Evals was launched. Our platform has a production-ready foundation with a significant amount of optimized features to maximize the user experience. Furthermore, it utilizes the latest technologies, and the project is fully open source to increase its overall lifetime. The end goal is to provide a platform that would improve the academic environment at Santa Clara University, satisfying the needs of students and professors alike.

After launching the platform, feedback was collected to iterate on the design and functionality to better suit the needs of our users. Once the changes were implemented, the platform was successfully relaunched with strong positive feedback. The volume of evaluations is still relatively small, but the platform is actively growing and continuously becoming more useful as data is collected. We believe further improvements to the platform are not necessary to achieve success, but it could help our users in various ways. Some of the suggested features include the ability for faculty to interact with students and an integration with the SCU registrar, if possible.

Table of Contents

List of Figures	vi
List of Tables	vi
1 Introduction	1
2 Requirements	2
2.1 Functional	2
2.2 Non-Functional	2
2.3 Design Constraints	3
3 Use Cases	4
4 Activity Diagrams	8
4.1 Student Activity Diagram	8
4.2 Professor Activity Diagram	8
4.3 Administrator Activity Diagram	8
5 Conceptual Model	10
6 Architectural Diagrams	15
6.1 Database	16
7 Technologies Used	17
7.1 Back End	17
7.2 Front End	18
7.3 Miscellaneous	19
8 Design Rationale	20
8.1 User Interface	20
8.2 Technologies Used	20
9 Testing	22
9.1 Internals	22
9.2 Alpha	22
9.3 Closed Beta	22
9.4 Omission of Open Beta	23
10 Risk Analysis	24
11 Development Timeline	25

12 Societal Issues	26
12.1 Ethical	26
12.2 Social	26
12.3 Usability	27
12.4 Lifelong Learning	27
12.5 Compassion	27
13 Final Result	28
14 Future Work	32
15 Project Assessment	34
15.1 Evaluation	34
15.2 Feedback	35
15.3 Metrics	36
16 Lessons Learned	38

List of Figures

1	Use cases diagram.	4
2	Student activity diagram.	9
3	Professor activity diagram.	9
4	Administrator activity diagram.	9
5	Home page.	11
6	Signed-in home page	11
7	Course evaluations.	12
8	Professor evaluations.	12
9	Post evaluation.	13
10	Evaluation form.	13
11	Administrator panel for managing evaluations.	14
12	Administrator panel for managing users.	14
13	Architectual diagram.	15
14	ER diagram.	16
15	Development timeline.	25
16	Signed out Homepage	28
17	Homepage	29
18	Profile	29
19	Post Evaluation	30
20	Browse Evaluations	30
21	Course Evaluations	31
22	Search	31
23	Acquisition Overview.	36
24	Sessions by Device.	37
25	Events	37

List of Tables

1	Table outlining the risks, consequences, and solutions for the project	24
---	--	----

1 Introduction

Students want to know the characteristics of the classes—and the professors teaching them—that are offered at their college institutions. Students also want this information at their fingertips. They want to go online, search for what they need, and then have it presented to them in an easy-to-read format. Furthermore, professors also want to get feedback from the students to find out how they can improve or change the class to make it a better learning experience.

The existing solutions to the problem lacks many basic features that we feel are essential to learning about courses and professors at college institutions. For example, Santa Clara University urges students to fill out official course evaluations at the end of each quarter. However, their evaluations are often difficult to sort through, and many of the questions asked are unclear and subjective to the student being surveyed. Usually, these evaluations are also overly complicated and the questions are not necessarily relevant to future students browsing the courses. This decreases student engagement which in turn lessens the benefit for all parties involved.

Another major solution that is not specific to any college is “RateMyProfessor.com” [16]. It is a third-party company that offers pre-structured surveys. Although this idea has potential, their solution is oversimplified, lacks many useful statistics, and offers no easy way of browsing courses. We believe that students should be able to view detailed statistics of previous evaluations, and “RateMyProfessor.com” only offers three viewable statistics, which is easiness, a total score average, and professor attractiveness. Due to the fact that it only offers surveys of professors, differentiating overall ratings between classes is also not possible. Additionally, anybody, including those not enrolled at the college, may write evaluations, making those specific evaluations useless.

Our solution focuses on creating a platform where students can evaluate professors and their classes in an objective and statistically meaningful way. We offer an intuitive interface that promotes student engagement in writing and reading course evaluations. Only faculty and students that are enrolled at an institution are able to participate in the platform in order to increase data quality. Evaluations are pre-structured, and ask questions that are easy for students to both understand and respond. Finally, browsing courses, professors, and evaluations is possible in ways that allow the user to quickly find what they need, by searching, filtering, and sorting the available data.

2 Requirements

The following requirements, divided into different categories, define the basic needs for the final product to be considered completed. The critical requirements are considered to be necessary, while the recommended requirements could be useful if there is time.

2.1 Functional

- Critical
 - It should allow the student to...
 - * Authenticate using their university email
 - * Post evaluations for a course/professor combination
 - * Read evaluations if they have posted an evaluation for the current quarter
 - * View evaluations for a specific professor
 - * View evaluations for a specific course
 - * Browse courses according to evaluation scores
 - * Browse professors according to evaluation scores
 - * Specify their majors, gender, and planned graduation year
 - * Vote on whether other evaluations are helpful or not
 - * See detailed statistics regarding course and professor evaluations
 - * View whether a professor has tenure
- Recommended
 - It is recommended that it allow...
 - * The professor to interact with the students
 - * The professor to have a manageable profile
 - * The professor to get notifications when new evaluations for it are posted
 - * The student to flag/report other students' evaluations

2.2 Non-Functional

- Critical

- It will be...
 - * Fast and responsive
 - * Easy and encouraging to use
 - * Easy to expand with new features
 - * Easy to manage as a platform administrator
 - * Easy to use on all kinds of platforms
- Recommended
 - It is recommended that it should be...
 - * A useful platform for both professors and students
 - * Modular and easy to expand to other universities
 - * Visually appealing and up-to-date with recent standards

2.3 Design Constraints

- Edge, Firefox, and Chrome versions are supported
- Deployable on Heroku
- Student and faculty verification

3 Use Cases

The system has three kinds of roles. There are students, professors, and administrators. As can be seen in figure 1, there are a few use cases that are shared between them, but some are also unique to one role. A user can have multiple roles. The administrator's main purpose is to manage users and evaluations. A system administrator would most likely have that role.

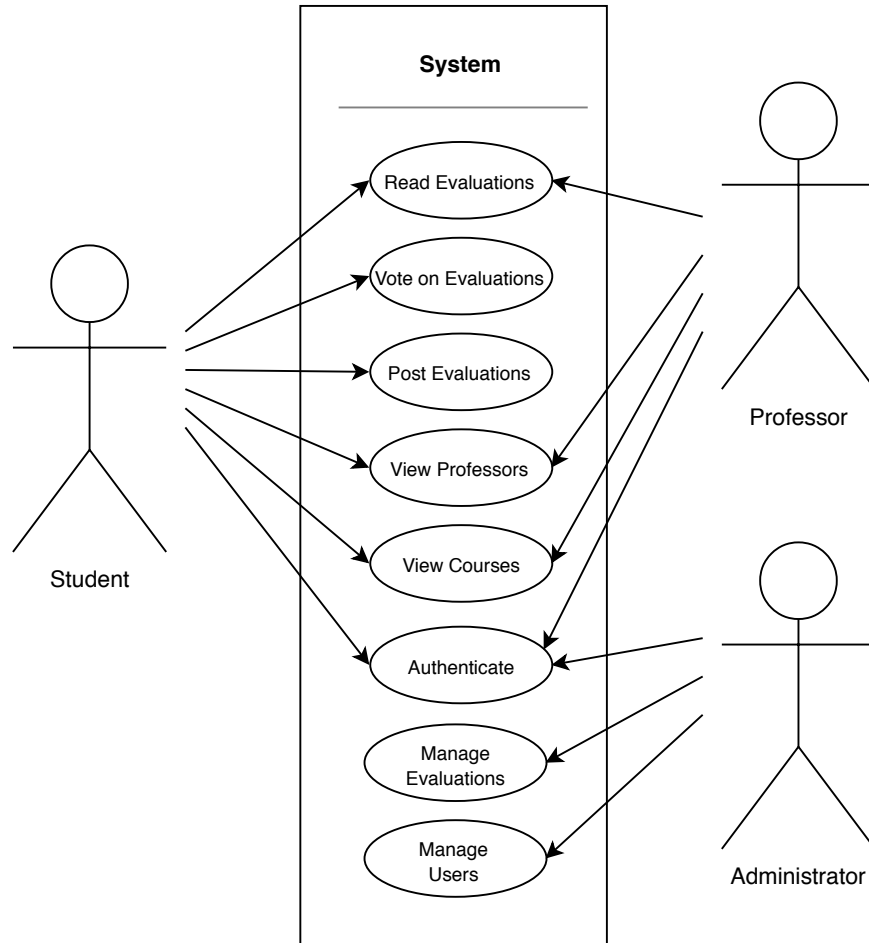


Figure 1: Use cases diagram.

- **Authenticate**

- **Goal:** Authenticate the user with the system and assign appropriate roles to actors.
- **Actors:** Student, Professor, Administrator
- **Preconditions:** Authenticate
- **Steps:**
 - a) Authenticate using email.
- **Postconditions:** The user is authenticated.
- **Exceptions:** The user is not allowed to access the platform.

- **Read Evaluations**

- **Goal:** Provide the actor with evaluations of professors and courses
- **Actors:** Student, Professor
- **Preconditions:** Authenticate
- **Steps:**
 - a) Search for a professor or course.
 - b) View evaluations for a professor or course.
- **Postconditions:** None
- **Exceptions:** None

- **Rate Evaluations**

- **Goal:** Allow the actor to rate evaluations.
- **Actors:** Student
- **Preconditions:** Authenticate, View evaluations for professors/courses.
- **Steps:**
 - a) Rate evaluation as helpful or unhelpful.
- **Postconditions:** The score for the evaluation is altered.
- **Exceptions:** None

- **Post Evaluations**

- **Goal:** Allow the actor to post an evaluation for a professor and course.

- **Actors:** Student
- **Preconditions:** Authenticate
- **Steps:**
 - a) Select the quarter the class was taken.
 - b) Select the course that was taught.
 - c) Select the professor that taught it.
 - d) Fill in and submit evaluation.
- **Postconditions:** An evaluation would be posted for the selected combination.
- **Exceptions:** The flow is interrupted by the user.

- **View Professors**
 - **Goal:** Allow the actor to view the evaluations for a professor.
 - **Actors:** Student, Professor
 - **Preconditions:** Authenticate
 - **Steps:**
 - a) Search for a professor.
 - b) View the evaluations for the selected professor.
 - **Postconditions:** None
 - **Exceptions:** None

- **View Courses**
 - **Goal:** Allow the actor to view the evaluations for a course.
 - **Actors:** Student, Professor
 - **Preconditions:** Authenticate
 - **Steps:**
 - a) Search for a course.
 - b) View the evaluations for the selected course.
 - **Postconditions:** None
 - **Exceptions:** None

- **Manage Evaluations**

- **Goal:** Allow the actor to manage evaluations.
- **Actors:** Administrator
- **Preconditions:** Authenticate
- **Steps:**
 - a) View evaluations.
 - b) Execute action.
- **Postconditions:** Evaluations could be deleted.
- **Exceptions:** None

- **Manage Users**

- **Goal:** Allow the actor to manage users.
- **Actors:** Administrator
- **Preconditions:** Authenticate.
- **Steps:**
 - a) View users.
 - b) Execute action.
- **Postconditions:** Users could be banned or granted administrator right.
- **Exceptions:** None

4 Activity Diagrams

The following diagrams represent the beginning of the flow using a black circle, and the end with a black circle containing a white ring near the outside border. The remainder of the notation is standard UML[3].

4.1 Student Activity Diagram

The Student activity diagram (figure 2) shows the options that are offered upon using the website application. Students begin by logging in using their associated SCU email. Then, depending if one wishes to read or write evaluations, one will either search for the professor or class directly and read the corresponding evaluations, or first select the quarter, then the course, and finally the professor before ultimately filling out and submitting the evaluation form. Then, if students wish to continue interacting with the website application, they can repeat this process until they are satisfied, when they may log out of the application.

4.2 Professor Activity Diagram

The Professor activity diagram (figure 3) shows the options that are offered upon using the website application. Professors begin by logging in using their associated SCU email. Then, to read evaluations one will either search for the professor or class directly and read the corresponding evaluations. Then, if professors wish to continue interacting with the website application, they can repeat this process until they are satisfied, when they may log out of the application.

4.3 Administrator Activity Diagram

The flow of administrators (figure 4) also starts with logging into the website application, where they may then choose to focus on evaluations or students. If focusing on evaluations, they may choose to delete an evaluation if deemed inappropriate or going against the posting rules. If focusing on students, the administrator can choose to ban a student from using the website application by blacklisting their email if the account has been deemed

malicious, abusive, or otherwise breaking posting rules. If administrators are finished, they may log out of the application.

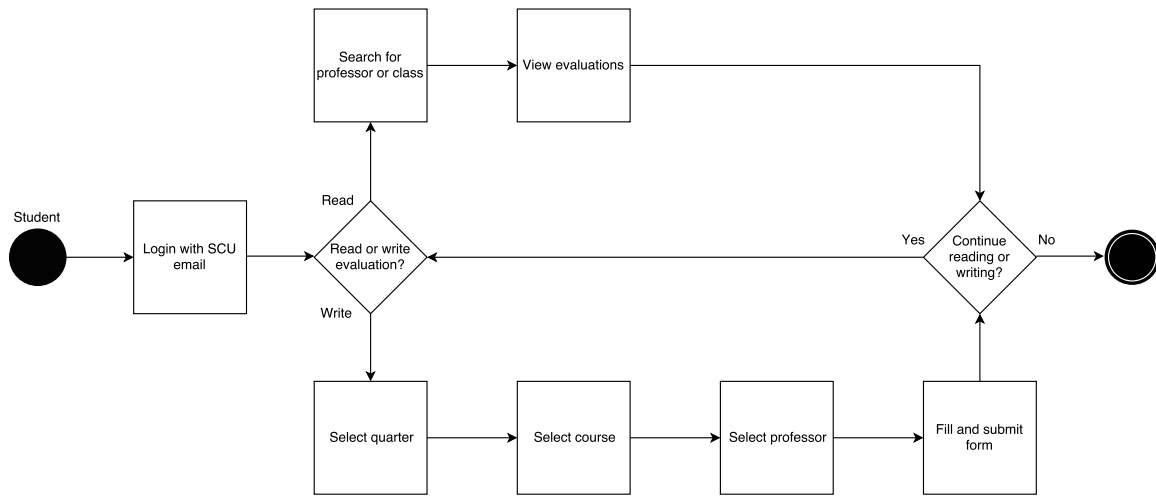


Figure 2: Student activity diagram.

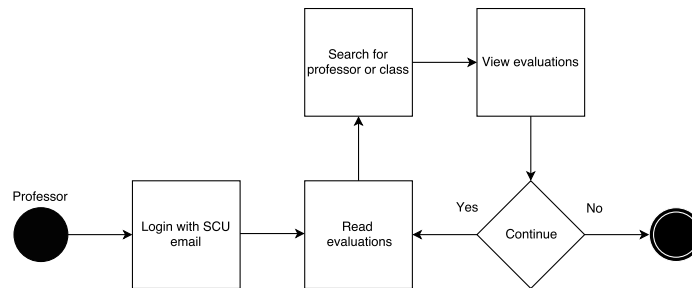


Figure 3: Professor activity diagram.

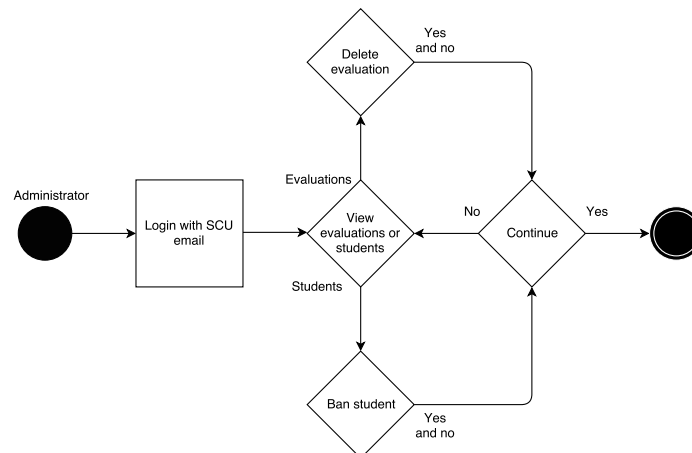


Figure 4: Administrator activity diagram.

5 Conceptual Model

The following figures, number 5-12, show the basic concept of the different main pages of the application. The home page (figure 5) only allows the user to authenticate with the application, since that is a requirement for using it. Once the user is authenticated, they can access the signed-in home page (figure 6) which will provide them with the basic actions available. Figure 7 and figure 8 show the interface that is presented to the user when they read evaluations about courses and professors. If the user decides to post an evaluation, they will be presented with the interface shown in figure 10, which will allow them to input all the necessary information for the evaluation and then submit it to the system. Finally, figure 11 and 12 show the administrator interface for managing evaluations and users respectively.

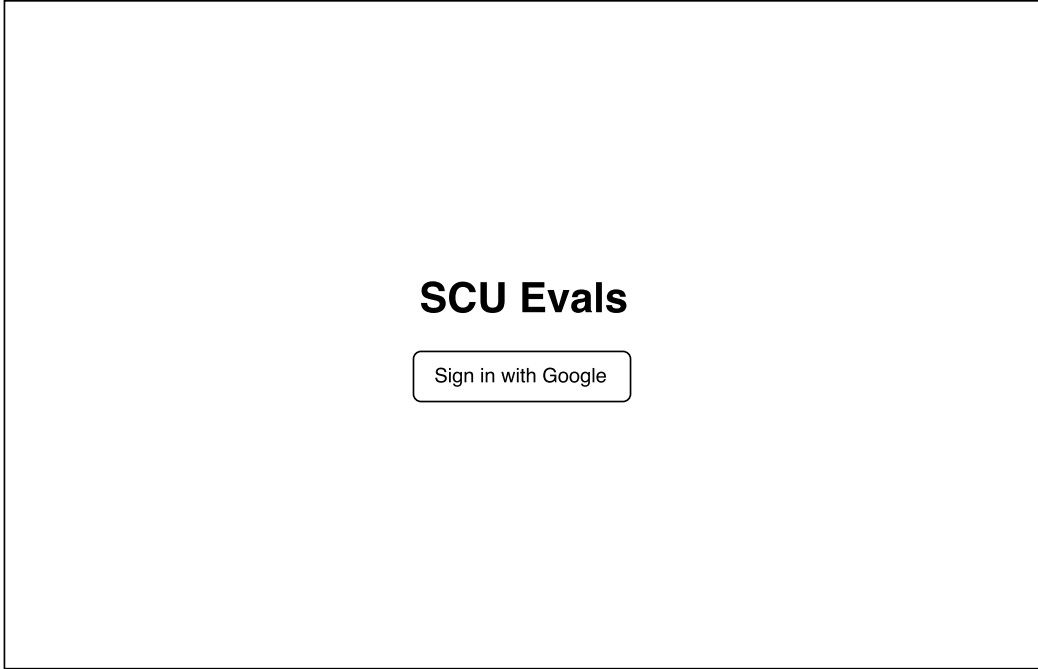


Figure 5: Home page.

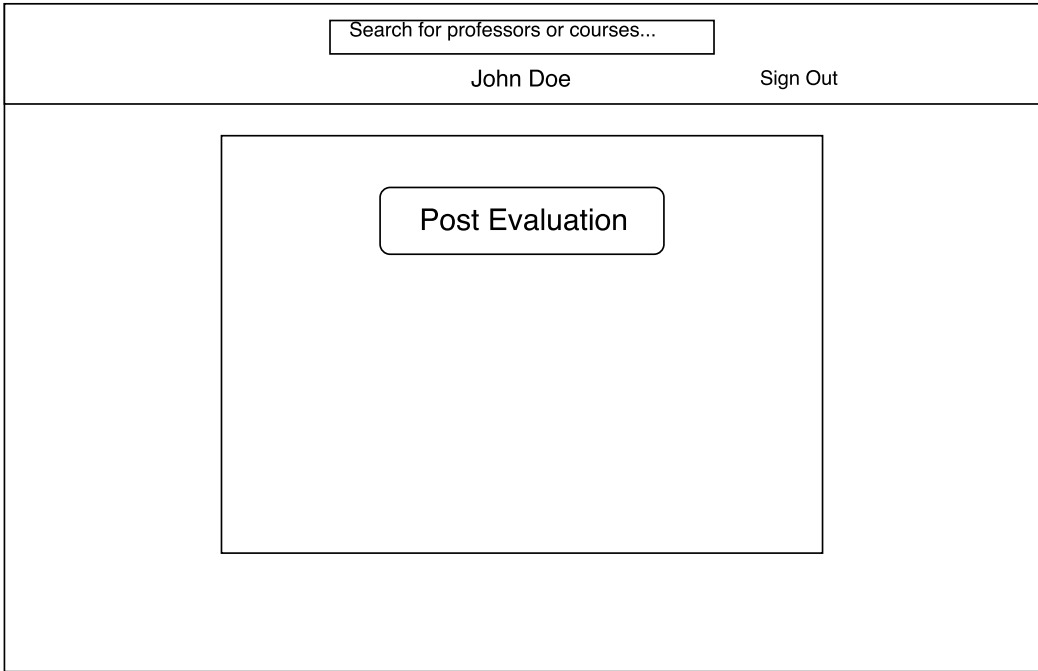


Figure 6: Signed-in home page

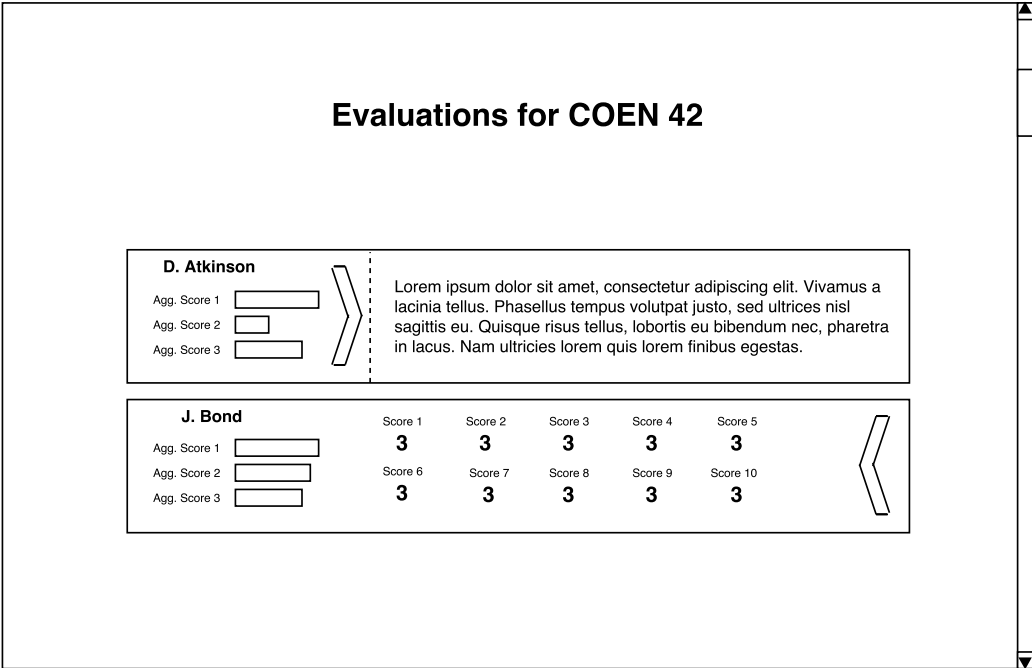


Figure 7: Course evaluations.

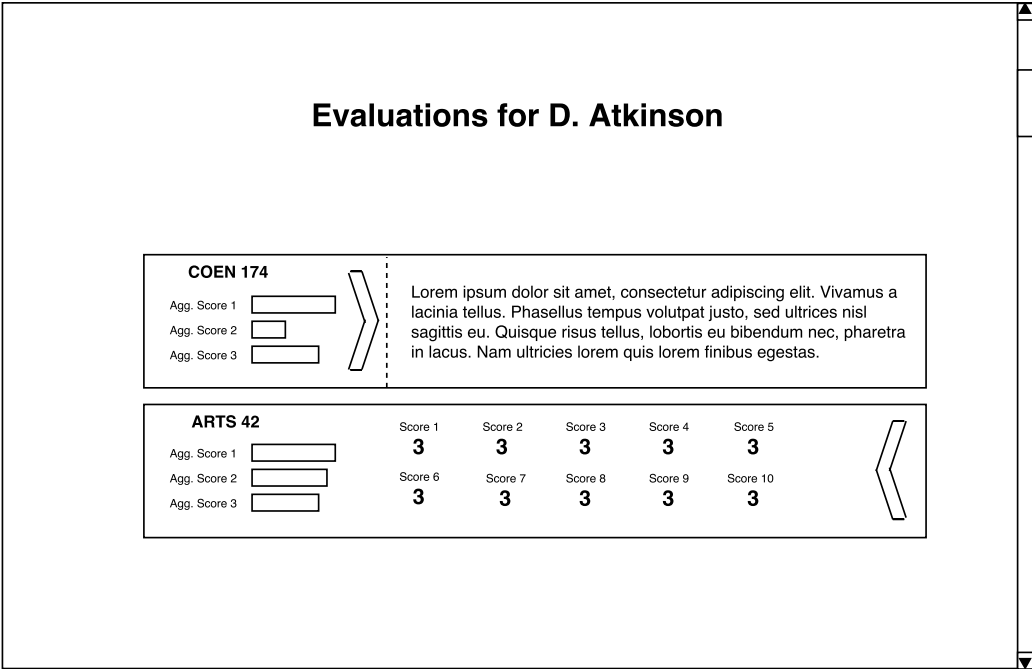
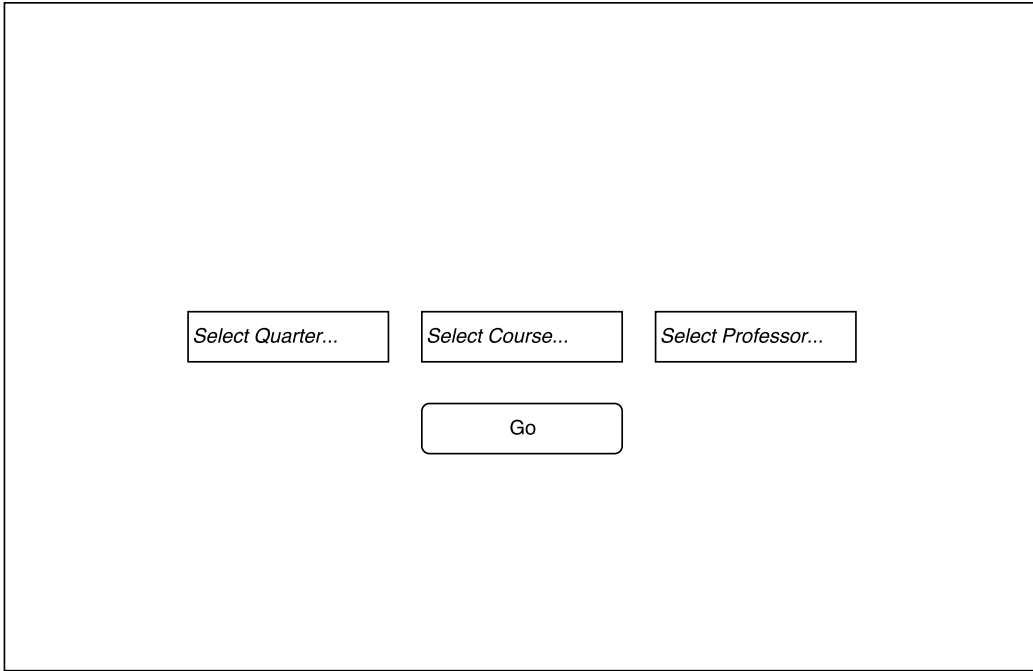
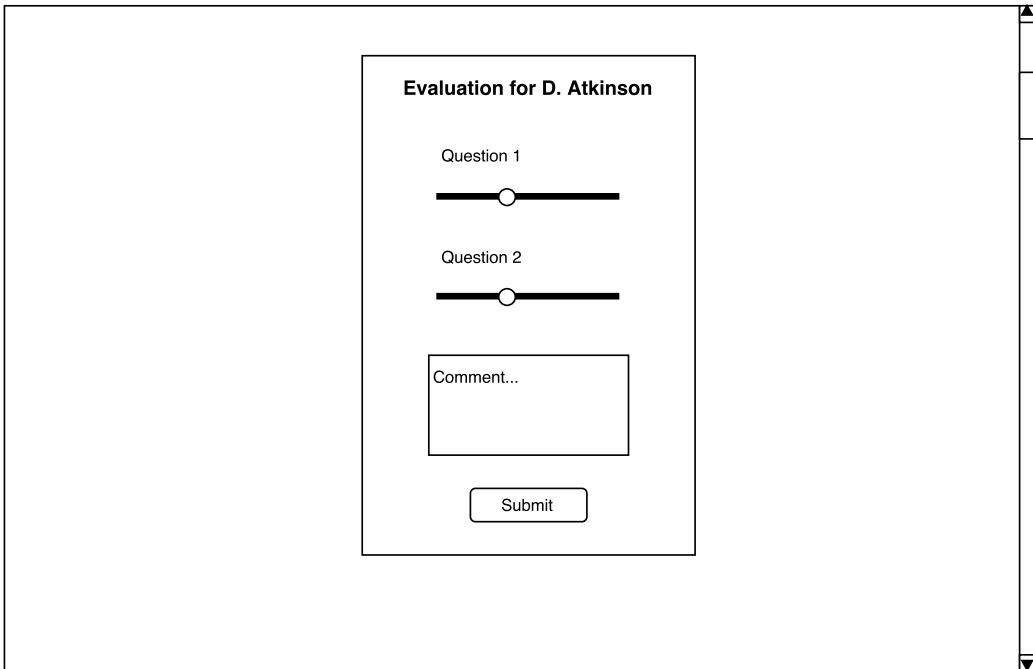


Figure 8: Professor evaluations.



A screenshot of a web interface for post-evaluation. It features three rectangular buttons arranged horizontally: "Select Quarter...", "Select Course...", and "Select Professor...". Below these buttons is a single rounded rectangular button labeled "Go".

Figure 9: Post evaluation.



A screenshot of an evaluation form titled "Evaluation for D. Atkinson". The form is contained within a central rectangular box. It includes two horizontal sliders, each labeled "Question 1" and "Question 2" respectively, with a small circle indicating the current rating. Below the sliders is a text input field labeled "Comment...". At the bottom of the form is a rounded rectangular "Submit" button. The entire form is set against a background with a vertical scrollbar on the right side.

Figure 10: Evaluation form.

Evaluations

Student	Professor	Course	Delete
Mary Jane	D. Atkinson	COEN 42	X
Peter Parker	D. Atkinson	COEN 42	X

Figure 11: Administrator panel for managing evaluations.

Users

Mary Jane	<input type="button" value="Make Admin"/>	<input type="button" value="Ban"/>
Peter Parker	<input type="button" value="Make Admin"/>	<input type="button" value="Ban"/>
John Doe	<input type="button" value="Remove Admin"/>	<input type="button" value="Ban"/>

Figure 12: Administrator panel for managing users.

6 Architectural Diagrams

Our architectural model is a hybrid model, based from data-centric [19] and client-server [19] architectural models. We made our model modular to easily abstract portions during development and to make portions of our program more independent and abstract. Users will interact directly with the front end server, which serves the HTML [7] layout. When requesting data, the back end receives AJAX [12] requests and fetches and returns the requested data from the database. This bi-directional flow of multiple servers helps lessen the load on each server .

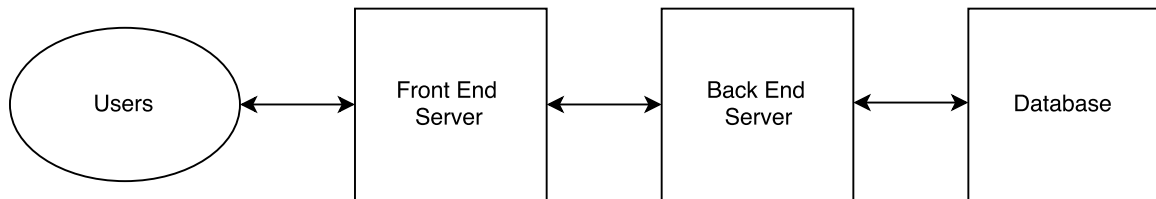


Figure 13: Architectural diagram.

6.1 Database

The diagram in figure 14 shows the ER model [6] that is used.

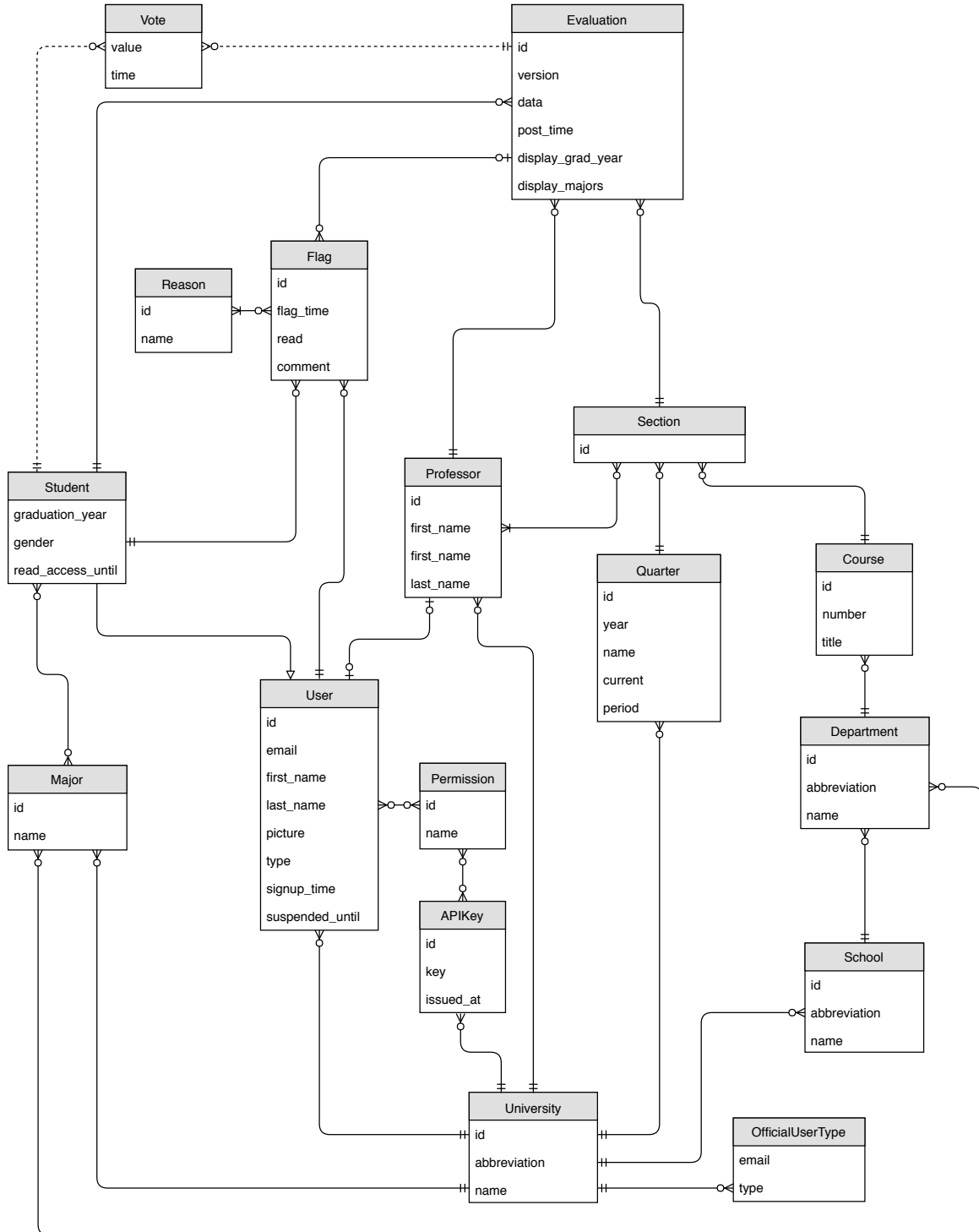


Figure 14: ER diagram.

7 Technologies Used

The following sections describe which technologies are used in the different parts of the application. The use of each technology is also briefly described.

7.1 Back End

- Python [11]
 - As a well-supported simple scripting language it allows for quick progress at the expense of extreme performance. However, it still provides enough performance for the intended use. It also allows us to run the project on multiple platforms without issues.
- Flask [9]
 - It is one of the major web application frameworks for Python that allows us to make most of the design decisions ourselves while providing us a reliable base. Furthermore, it has plenty of extensions available that enabled us to achieve more in less time.
- PostgreSQL [15]
 - A relational database fits the project's needs due nature of the data. This one is one of the most popular ones, with good support and a wide variety of features. It is also well-supported by many hosting providers.
- npm [21]
 - npm, or node package manager, allows easy package management for our project. It ensures that the user running the project that launches the server uses the correct versions of packages that we declare fit for our website application.
- Webpack [10]
 - It bundles modules together, making Babel possible as well as asynchronous loading of the transpiled code that Babel creates. It also provides a development server which updates live on changes that makes creating the website application much faster.

- Babel [1]
 - Making ES6 JavaScript compatible with older browsers requires the transpilation of Javascript into older versions. This makes cross-browser compatibility for JavaScript possible.

7.2 Front End

- JavaScript [18]
 - This is essential for our platform since we are running React. ES6 standards such as classes, importing packages, and using newly introduced shortcuts, are implemented.
- React [20]
 - The library ensures fast responses when interacting with the UI because of the virtual DOM and preservation of HTTP requests. It makes development fast because it offers a modular solution for development that abstracts components from each other.
- Redux [17]
 - Managing states becomes much easier with a central location that can offer connection to React components which then become containers. It makes React containers re-render upon state changes in the store it creates.
- Sass [5]
 - Sass compiles .sass to .css, and allows programmatic creation of stylesheets that allow easy cross-browser compatibility, variables, mixins, and more. It also allows compressed creation of stylesheets.
- Bootstrap [8]
 - This allows flexible, responsive design while making development easy and fast.

7.3 Miscellaneous

- GitHub [2]
 - Source control is a given for any serious project and GitHub is a good choice due to its intuitive interface and good project management for multiple contributors. It also allows us to remotely access our code in an easy fashion.
- OAuth2 [4]
 - To be able to verify that only students and faculty of the selected university will be able to interact with our application, we use OAuth2 authentication through Google's servers. It minimizes security risks since we do not have to store passwords in any way, and makes it easier for the end-user since they do not have to go through a lengthy account-creation process.
- JSON Web Tokens [13]
 - This technology was chosen over normal cookie/session authentication due to its flexibility and simplicity. Instead of making unnecessary database queries every time a user has to be authenticated, we can utilize the secure signing of JSON web tokens to instantly validate whether a user is authorized or not.

8 Design Rationale

For developing the project, we chose our design based on the criteria that we wished to fulfill. Below, we elaborate on the reasons for these decisions.

8.1 User Interface

Our UI was designed as a single page application that can be used on any device with browser capability. We wanted our application to be as simple as possible without compromising performance or usability.

- We wanted our website application to be mobile friendly with a flexible browser design based on pixel widths of devices. We chose this because we were aware that most of our users own multiple devices with different screen size constraints, and we wish to allow them to easily use our website application with each device.
- Our page interaction relies on AJAX requests to the back end, meaning that page reloads are not necessary once inside the website application. Thus, our product feels fluid and responsive without page load interruptions.
- Base colors and designs were chosen and reused throughout our application to easily recognize various element such as buttons, links, and paragraphs. Sass helped us with this by allowing us to create reusable variables.

8.2 Technologies Used

- The choice fell on Python combined with Flask and PostgreSQL for the back end due to them being a well-tested and well-supported trio. The number of libraries available for interaction between these components is big which makes development faster and easier. Furthermore, these technologies are widely supported by different hosting companies, which makes it easier for us to move the application from one provider to another if necessary.
- We chose to use a package manager for easily managing our libraries and their dependencies. It also ensures that the correct versions of libraries are being used based

on our declared rules for the manager. npm, or node package manager, is currently one of the most popular package managers, and is included with Node, which with we run our server.

- For cross-browser compatibility, we chose Sass, Bootstrap, and Babel to help achieve our criteria. Sass compiles our CSS to be cross-browser compatible and fast for programmatic creation of CSS. Bootstrap lets us achieve our flexible design criteria while maintaining a modern design that is fast to develop. Babel transpiles our Javascript into cross-compatible Javascript so we may use the latest features of the language for browsers that do not yet support them as well as older versions of browsers.
- React and Redux allow easy, efficient re-rendering of HTML components. React smartly detects what needs to be re-rendered based on states. Most of these states are stored in a central accessible store with Redux. These states act as event handlers when changed to re-render components listening to specific states to update the information when needed. This allows the efficient manipulation of DOM and single page applications with AJAX incorporated.

9 Testing

The following sections describe the different phases of testing that were conducted for the platform.

9.1 Internals

To make sure that the mechanics of the application ran well in all the required scenarios, we employed an extensive internal black-box testing suite that is run regularly and before every new version of the application was released. Furthermore, we closely monitored the testing coverage to be certain that all use cases were included in the testing suite. This whole process minimized the amount of bugs and other flaws that reached the end-user and simplified debugging and development.

9.2 Alpha

Once we deemed the platform fully functional, we started our alpha testing. It was a process in which we ourselves went through the different flows of the application, making sure that no issues arose. Any issues that did arise were fixed before we entered the next phase. We also made sure that all the requirements and design constraints were fulfilled at this stage.

9.3 Closed Beta

After the alpha testing concluded and was successful, we initiated the closed beta phase. It consisted of allowing a limited number of trusted users onto the platform, where they could roam around freely, as well as complete a list of tasks that we assigned them. Once they had done that, we collected user feedback on all parts of the application. This gave us a good picture of what had to be changed in order to make the application more user-friendly.

9.4 Omission of Open Beta

The decision was made to not have an open beta, and instead go directly to a production-ready application once the closed beta phase concluded. The motivation was that there was not enough time for us to make any drastic changes to the application once it launched, and therefore it would not be justified to first have a beta phase. If a beta phase were to be had, it is likely that the application would never have left it.

10 Risk Analysis

Naturally, a project always comes with certain risks. We identified the major ones in table 1 along with how we mitigated them. Severity is judged on a 10-point scale, and the impact score is calculated by multiplying the probability with the severity.

Table 1: Table outlining the risks, consequences, and solutions for the project

Risk	Consequences	Probability	Severity	Impact	Mitigation
Small user-base	Lacking data, not many reviews	0.60	9	5.40	Frequently collect user feedback from a wide variety of users
Development delays	Missing features, less appealing interface, bugs	0.40	9	3.60	Start early and work during spare time
Browser compatibility issues	Hours of debugging and rewriting code	0.30	6	1.80	Use SASS to compile CSS and use Babel to transpile JS
Security flaws	Invalid form submissions, bad data	0.25	7	1.75	Verify identity each request, use JWTs
Unintuitive UI	Fewer users, less user interaction, less useful platform	0.35	5	1.75	Apply common UI standards, research popular designs

11 Development Timeline

To make sure we could deliver the application on time, we laid out a development timeline in figure 15. The green boxes indicate when we worked on the different parts. Some parts had to be worked on simultaneously due to their functionality being intertwined. Early on in the work-flow there was no real priority.



Figure 15: Development timeline.

12 Societal Issues

All engineering projects tend to revolve around several different societal issues, which are important to discuss in order to make it clear that the developers have the right intentions and motivations. The sections that follow discuss the issues that we find relevant to this project. We believe that our project does not touch upon Political, Economic, Health and Safety, Manufacturability, Sustainability, and Environmental Impact issues.

12.1 Ethical

One of the goals with our platform is to keep the students anonymous at all times. This is to allow everyone to feel safe no matter if they want to criticize or praise a professor or course. It is therefore of utmost importance that we handle the data in a secure and anonymous fashion. While names and emails are stored unencrypted in the database, the back-end never transmits this information in relation to evaluations or votes. Therefore, unless someone gets unauthorized access to the database, all users will remain anonymous to each other.

Additionally, we implemented a flagging system so students may flag evaluations if they break our guidelines or are otherwise inappropriate. With this system, we are alerted of the report and may take action to ensure our platform is used ethically, including removing the evaluation or banning a user if necessary.

12.2 Social

As a platform for students to evaluate their professors, it would naturally have an impact on the community that these professors and students are a part of. The goal is for the impact to be a positive one, where students can give useful feedback to professors that can then learn and adapt to improve the experience for everyone. What could happen is that some students could write harsh evaluations that impact professors' mental health in a negative way. The way to prevent this is to work hard on building a mutual understanding of the pros and cons of such evaluations.

12.3 Usability

When we started designing this platform we wanted to make sure that it had an intuitive interface so that any student or faculty could get the hang of it as quickly as possible without any extra help. We believe this is something we were able to achieve after collecting feedback. Furthermore, the platform works well on all kinds of screen sizes, which is another important aspect of usability nowadays.

12.4 Lifelong Learning

This project certainly encouraged both of us to explore new technologies that we had not yet mastered. By spending countless of hours on going through tutorials, documentation, and examples, we improved our self-teaching skills, which will come in handy in the future.

12.5 Compassion

A big part of why this platform came to be was the compassion we felt towards our fellow students that have had to put up with professors who could have used a lot more feedback than they were given. College is an incredible financial burden for a lot of students, which is why it is of great importance that said students get value for their money. By building a platform that helps them help their professors, we hope to improve the overall quality of the education that is offered at their university.

13 Final Result

In this section, some of the major components of the website application are displayed. This section will not cover all of the functionality of the site, but is only meant as a general idea of the scope of the project.

Figure 16 shows the signed out page of SCU Evals. There, the user is prompted to sign in with their SCU google account.



Figure 16: Signed out Homepage

In figure 17, the signed in homepage shows both recent evaluations and a top list for both courses and professors, sorted by default by your major (or if undeclared, then by all). There is also easy access to browse or post evaluations.

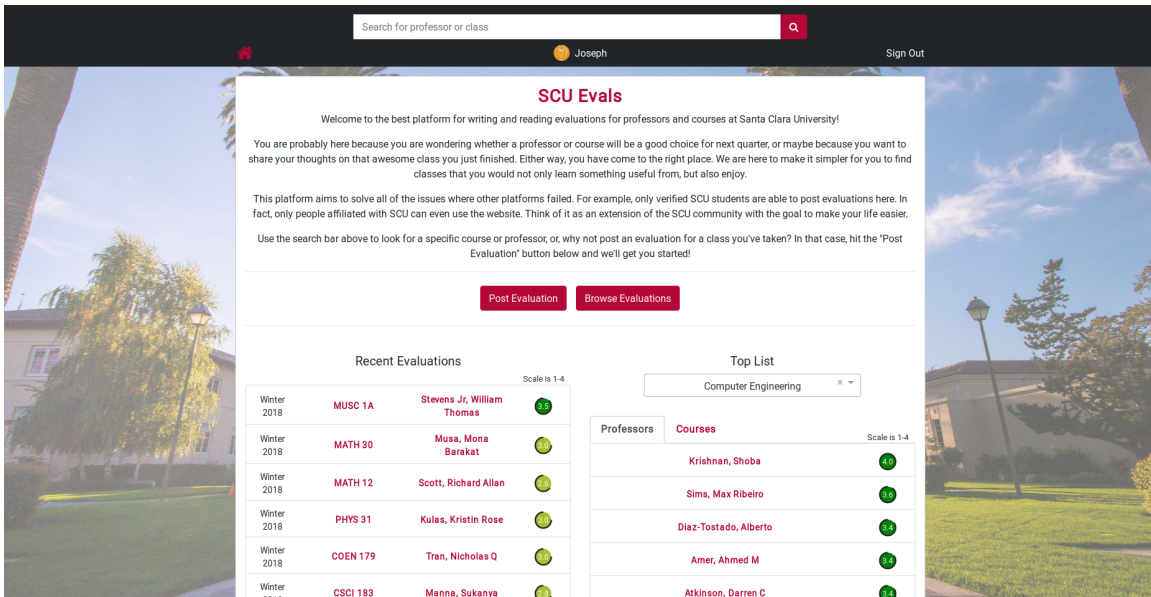


Figure 17: Homepage

The profile page is clean and easy to update. We give users the ability to update all of their info they share with us when completing their profile. This is shown in figure 18.

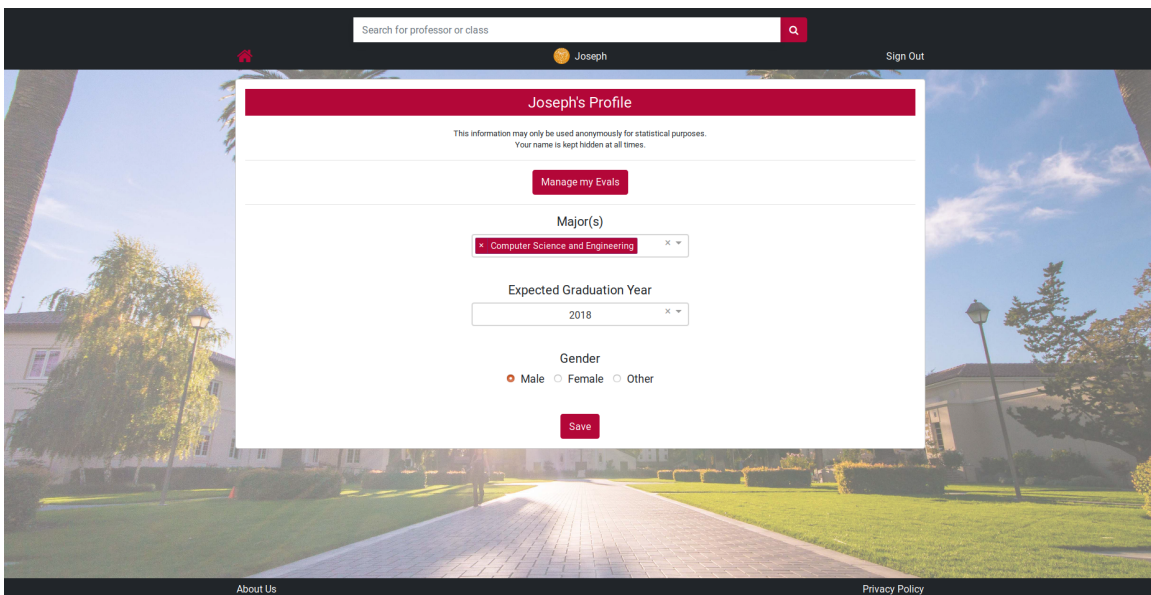


Figure 18: Profile

Figure 9 shows how students are able to post on the platform. As the user changes scores, they are given prompts for each question to help direct the user. Additionally, if the user finds a field unclear, they can hover over the nearby question icon to get a more detailed

description.

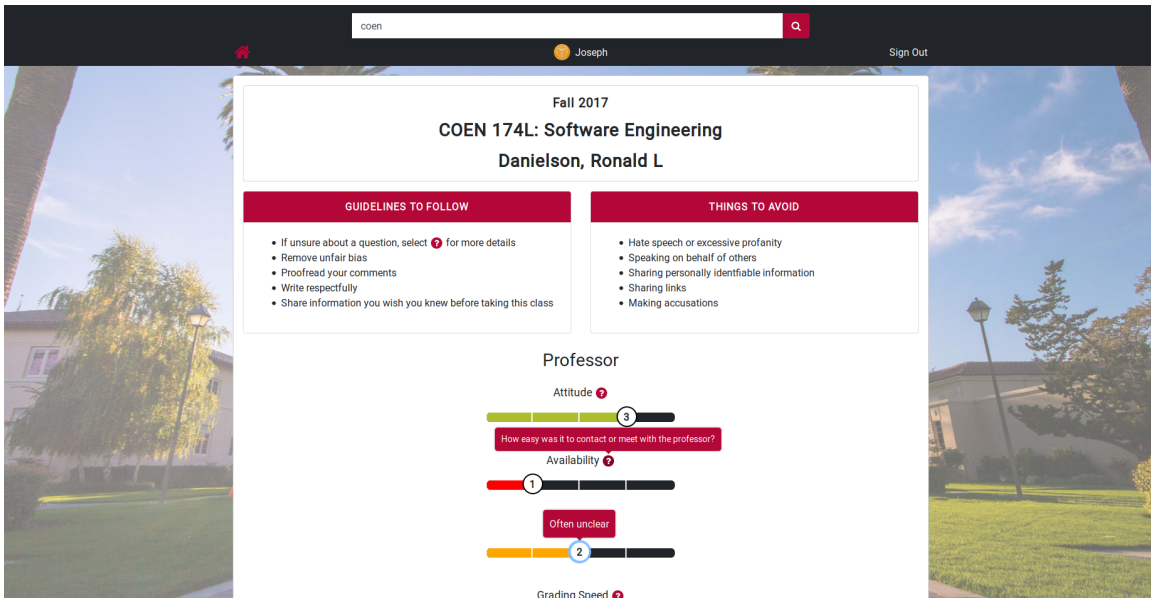


Figure 19: Post Evaluation

Browsing evaluations allows users to filter and sort by different fields. As filters are altered, the average scores are updated. This gives users an efficient way of finding specific evaluations. In figure 20, we added a filter to only show evaluations for Spring 2017 and sorted them by courses as an example.

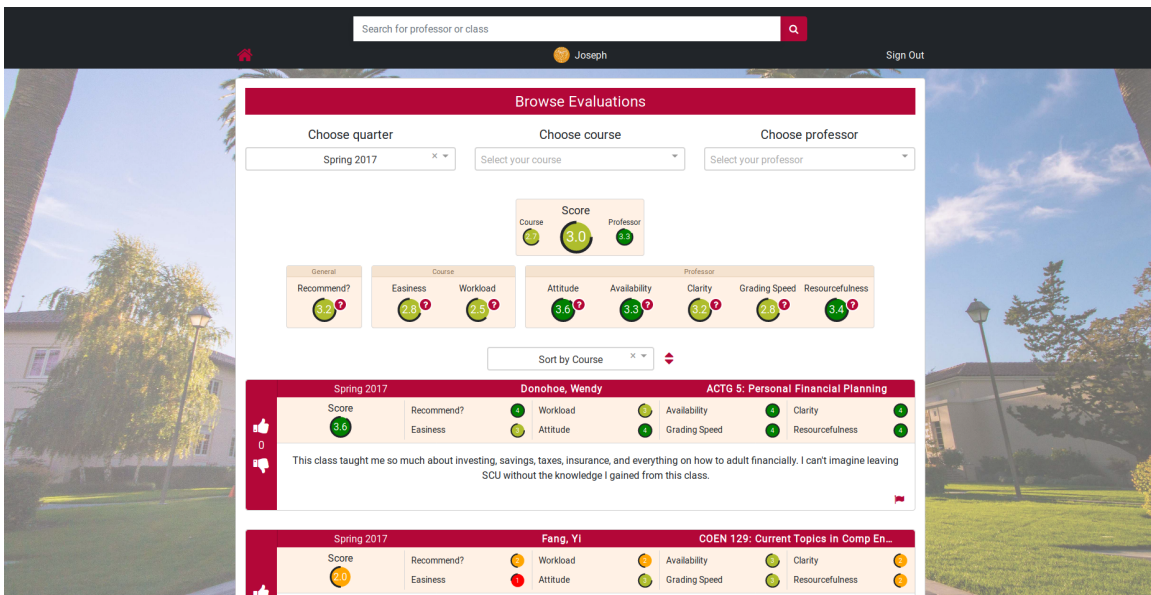


Figure 20: Browse Evaluations

As well as browsing evaluations, users can choose to view a page specifically for one professor or course. In figure 21, a course page is shown as an example. Viewing a professor page is practically the same, so no additional figure is needed.

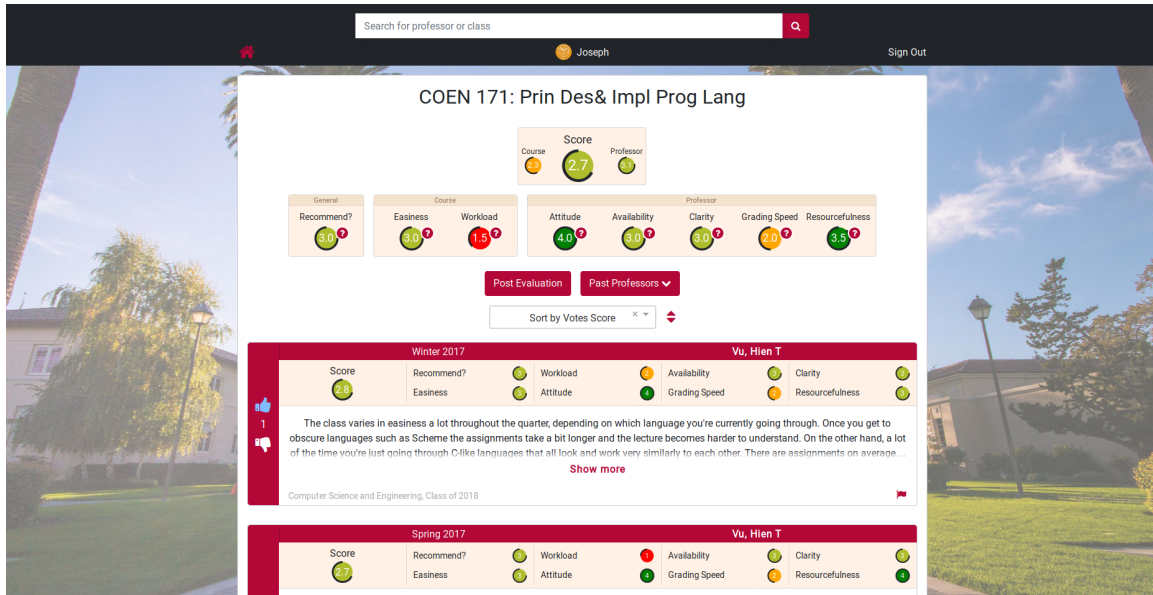


Figure 21: Course Evaluations

Users can also go to a search page that shows courses or professors that match their search results. It is not shown but should be noted that typing in the search bar also drops down results in a mini dropdown menu.

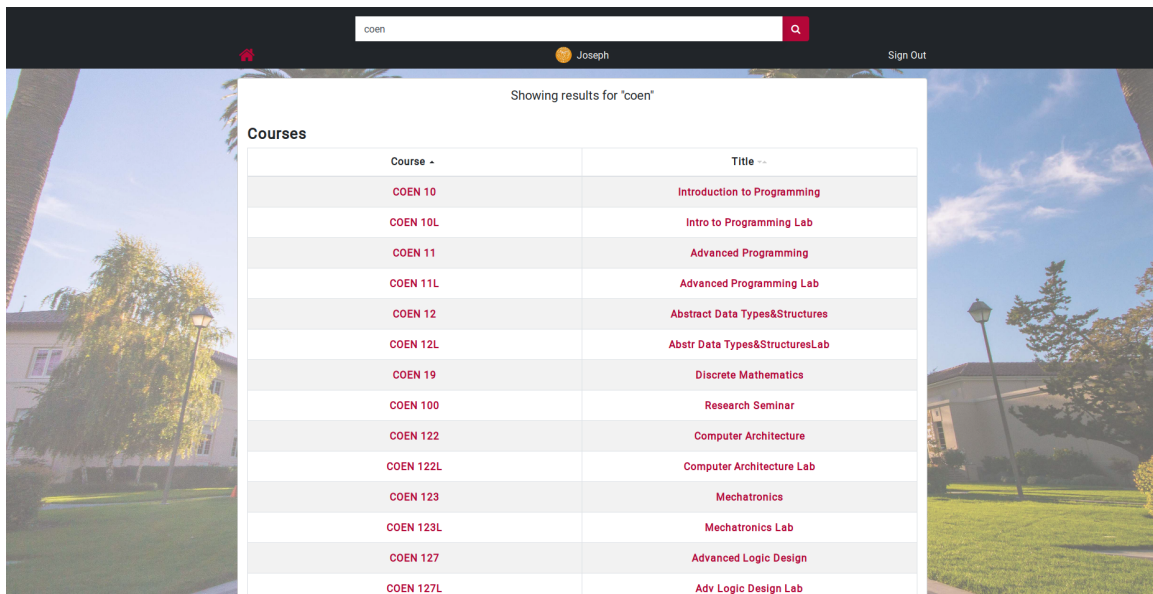


Figure 22: Search

14 Future Work

- Run indefinitely after graduation
 - Self-funded at first, with potential for revenue or break-even in the future
- Faculty responses to evaluations
 - There is currently no way for the faculty to interact with the students. By adding the ability for faculty to respond to individual evaluations it will be possible for the faculty to answer questions and/or criticism to further clarify different situations and to promote 2-way communication. This is something that other platforms lack completely.
- Add error tracking for the front end
 - Currently, when errors occur in the console, there is no way for us as developers to see when users have encountered one of these bugs. In the future, we will be adding a tool that reports any console errors to a service that allows us to track errors and ultimately fix them.
- Explore more ways to view evaluations data
 - As we collect more data, we can do more things with it including adding charts, graphs, advanced filtering options, etc. This is important because our platform is designed for users to heavily rely on this data, and offering more options could prove useful and time-worthy if the platform continues to grow.
- Fully automate data collection/scraping
 - At the moment the data scraper/collector still needs to be run manually after every quarter to ensure that the platform has the latest courses, professors, and majors. By increasing the reliability and fully automating it on a schedule, the platform would be even more self-sufficient and future-proof.
- Further incentivization of participation
 - While we already force users to post an evaluation for each quarter that they want access, more could be done. Students could accumulate “points” when their evaluations get voted up, and every time they post an evaluation. These points could in turn be shown off on leaderboards (anonymous or not) as well

as lead to other events (random raffles, their evaluations being marked with a quality badge of some sorts, etc.). This would encourage students to post more than just one evaluation and to vote on their fellow students' evaluations.

- Integration with official registrar systems
 - It is currently possible to validate who is a student and who is a faculty, but the platform does not have the ability to verify that student x actually took class y. The only reasonable way to achieve this would be through a direct integration with SCU's official registrar. It should be explored whether the school would be interested in providing this data, free of charge of course.
- Further generalization of code and theme
 - One of the original ideas with this platform was for it to be easily adaptable so that other schools could either run their own version of it, or share one with other schools. The code is very modular as is, it would not require a lot of work to customize it to fit a different university. However, it would be even better if the code did not have to be modified. Instead, there could be a configuration file where attributes such as university name and logo could be entered, that the rest of the system would then read in. Then, the only major part left that would inevitably have to be modified is the data scraper.

15 Project Assessment

The following sections provide an objective evaluation of the entire project, description of and response to the feedback we collected, and finally an analysis of different metrics surrounding the launched website.

15.1 Evaluation

The first public launch of SCU Evals occurred in the beginning of February 2018, during the registration period for spring quarter classes. The initial version had all the required functionality as stated by our design report, with the exception of faculty access. It was left out of the initial release due to prioritization of student functionality. If a faculty tried to sign in, they would be greeted by a message saying that they would get access in the near future. The launch was accompanied by an ad campaign on the Facebook/Instagram social network. Overall the initial launch went well and the initial response from users was positive.

After collecting feedback and improving the platform a launch of a new version was done in the beginning of May 2018, before the registration period for fall quarter classes and during senior design conference week. This launch was also accompanied by an ad campaign on Facebook/Instagram, as well as direct marketing to select SCU student groups on those networks. The decision was made to temporarily no longer give free reading access for the first quarter that a student signs up during. This was to try to increase the volume of evaluations that we would get, in order to make the platform more useful and consequently more attractive in the future. The plan worked out really well and many more evaluations were written than expected.

The changes/improvements between the first and the second launch were much needed and allowed the project to succeed more than if only one major version would have been launched. Collecting feedback was as expected clearly very helpful. Overall we are satisfied with how the project went, and all the positive response we got so far. Attracting faculty has been the hardest aspect, and one that still needs work in terms of marketing. Most faculty seemingly do not know about the platform yet. During the coming couple of years, the volume of data available on the platform should increase significantly (due to an expected constant increase in users), which we hope will be enough to organically attract

more faculty to take advantage of all the free feedback that the students give them.

15.2 Feedback

Feedback has been continually collected since January 2018, with more focused efforts right before and after the initial launch. Several students with different majors were told to walk through the platform while in front of us, and to describe their thoughts and ideas out loud in the meantime. Afterwards, they were briefly interviewed about their experience. Out of all the feedback we got, there were some parts that we, after having thorough discussions about, decided not to listen to/implement. Likewise, there was feedback that we did end up listening to/implementing.

Most of the changes that were the result of feedback were small UI changes across all of the platform. Users were sometimes confused about specific wording, they wanted more explicit instructions, and overall more clarifications. The most major change was to the two questions that surround the course, when writing an evaluation. Initially, they were the inverse of all the other questions (a score of 4 was the worst and 1 was the best). However, this caused major confusion when reading evaluations, therefore, they were changed to fit with the rest of the questions (with a score of 4 being the best, and 1 being the worst).

There were two major features that most of the students that were interviewed requested. The first feature was to allow students to input an exact number of hours that they spent on an average week in a class, instead of picking a score between 1 and 4. The students said that a more precise number would be more helpful than just a grade. However, we decided to not follow through with this idea after coming to the conclusion that it is often very hard for students to actually estimate how much time they spent on average. It tends to be a very subjective number due to a variety of factors. One of them being that a lot of courses have a very varying workload throughout the quarter, and another one being that coming up with a weekly estimate out of the blue is generally very hard since few people accurately remember how many hours they spent doing actual work, and not just socializing or being distracted.

The second highly requested feature was to allow students to input their final grade that they got in the class that they are evaluating. We decided not to implement this either due to several reasons. Partially, it is to be expected that students would generally only want to share their grade if they got a good one, and partially it is also to be expected that not a lot

of students would actually want to share their grade, since it can make it a lot easier for a professor to identify them, which tends to not be desirable. Due to the bias in who would share their grade, it would lead to misleading data. There is a good chance that students would quickly look at what grades people have shared, and then take for granted that they would receive a similar grade, ignoring all the other data that would most likely be more helpful to them since it is more objective.

15.3 Metrics

In figure 23, we see that from February 1 to May 30, 2018, the majority of traffic came from direct access to the site, with links from social websites coming in a close second. The advertising we ran on Facebook and Instagram should be accountable for this.

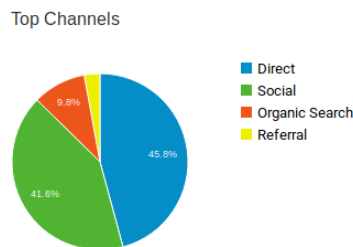


Figure 23: Acquisition Overview.

The majority of sessions are via desktop, but a large mobile presence can also be seen. Figure 24 shows the percentage of sessions categorized by kind of device. As seen, tablets are rarely used. This data is vital to realize that continued mobile development is relevant to our users.

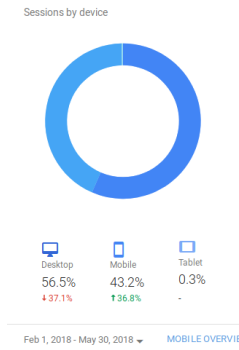


Figure 24: Sessions by Device.

We track six different events on our website, which are: signing up, completing the profile setup, submitting an evaluation, deleting an evaluation, voting for an evaluation, and removing a vote from an evaluation. Figure 25 shows the amount of evaluations that were posted weekly from February 1 to May 30, 2018.



Figure 25: Events

16 Lessons Learned

Outdated browser support is tedious and often not worth the time and effort when our target audience typically uses the most up-to-date versions of browsers. According to Google Analytics [14], we were able to support the browser choices for practically our entire audience; therefore, we found it wasteful to debug old browsers, and instead added tools that helped us automate this feature. Though not perfect, we believe this choice was the right decision.

We also found that project management is very time consuming and that it sometimes requires more effort than doing actual work on the project. For example, a lot of time went into restructuring code and ensuring correct code formatting in order to improve readability. Furthermore, general communication about ideas and plans also took up quite some time.

References

- [1] *Babel: The Compiler for Writing Next Generation JavaScript*. URL: <https://babeljs.io/>.
- [2] Peter Bell and Brent Beer. *Introducing GitHub: A Non-Technical Guide*. OReilly, 2015.
- [3] Grady Booch, James Rumbaugh, and Ivar Jacobson. *Unified Modeling Language User Guide, The, Second Edition*. Addison Wesley Professional, 2005.
- [4] Ryan Boyd. *Getting Started with OAuth 2.0*. OReilly Media, 2012.
- [5] Hampton Catlin and Michael Lintorn Catlin. *Pragmatic Guide to Sass 3: Tame the Modern Style Sheet*. The Pragmatic Bookshelf, 2016.
- [6] Peter Pin-shan Chen. “The Entity-Relationship Model: Toward a Unified View of Data”. In: *ACM Transactions on Database Systems* 1 (1976), pp. 9–36.
- [7] Jon Duckett. *HTML CSS: design and build websites*. John Wiley Sons, Inc., 2015.
- [8] Ben Frain. *Responsive Web Design with HTML5 and CSS3*. Packt Publishing Limited, 2015.
- [9] Miguel Grinberg. *Flask Web Development*. OReilly, 2018.
- [10] *Guides*. URL: <https://webpack.js.org/guides/>.
- [11] John V. Guttag. *Introduction to Computation and Programming using Python: With Application to Understanding Data*. MIT Press, 2016.
- [12] Anthony T. Holdener. *Ajax: The Definitive Guide*. OReilly, 2011.
- [13] M. Jones, J. Bradley, and N. Sakimura. *JSON Web Token (JWT)*. May 2015. URL: <https://tools.ietf.org/html/rfc7519>.
- [14] Alexa L. Mokalis and Joel Davis. *Google Analytics Demystified*. CreateSpace Independent Publishing Platform, 2018.
- [15] Regina O. Obe and Leo S. Hsu. *PostgreSQL: Up and Running: A practical guide to the advanced open source database*. OReilly, 2018.
- [16] *RateMyProfessors.com - Find and rate your professor or campus*. URL: <http://www.ratemyprofessors.com/>.
- [17] *Read Me - Redux*. URL: <https://redux.js.org/introduction>.
- [18] *Standard ECMA-262*. URL: <https://www.ecma-international.org/publications/standards/Ecma-262.htm>.
- [19] Frank F. Tsui, Orlando Karam, and Barbara Bernal. *Essentials of Software Engineering*. 4th ed. Jones Bartlett Learning, 2018.
- [20] *Tutorial: Intro To React - React*. URL: <https://reactjs.org/tutorial/tutorial.html>.
- [21] *What is npm? — npm Documentation*. URL: <https://docs.npmjs.com/>.