CSCI 370 Final Report

CSM EDNS 2: CS@Mines Oral Presentations 2.0

Team: One (1) Ramblin' Wreck

Michael Kormishkin
Thomas Lowery
Alexandra Cooke
Gabriel Del Castillo

Revised June 15, 2022

CSCI 370 Summer 2022

Dr. Rob Thompson

Table 1: Revision History

Revision	Date	Comments
New	May 20, 2022	Completed Sections: I. Introduction II. Functional Requirements III. Non-functional Requirements IV. Risks V. Definition of Done VI. XI. Team Profile References Appendix A - Key Terms
Rev - 2	May 26, 2022	Updated Sections: I. Introduction II. Functional Requirements III. Non-functional Requirements IV. Risks V. Definition of Done Completed Sections: VI. System Architecture
Rev – 3	June 3, 2022	Updated Sections: VI. System Architecture Completed Sections: VII. Software Test and Quality VIII. Ethical Considerations
Rev - 4	June 10, 2022	Updated Sections: I. Introduction VII. Software Test and Quality
Rev - 5	June 15, 2022	Completed Sections: VII. Technical Design X. Project Completion Status XI. Future Work XII. Lessons Learned Updated Sections: II. Functional Requirements III. Non-Functional Requirements IV. Risks V. Definition of Done VI. System Architecture VIII. Software Test and Quality IX. Project Ethical Considerations

Table of Contents

I. Introduction	2
II. Functional Requirements	2-3
III. Non-Functional Requirements	3
IV. Risks	3
V. Definition of Done	4
VI. System Architecture	4-8
VII. Technical Design	8-9
VIII. Software Test and Quality	9-10
IX. Project Ethical Considerations	10-11
X. Project Completion Status	12
XI. Future Work	12-13
XII. Lessons Learned	13-14
XIII. Team Profile	14-15
Appendix A – Key Terms	15

I. Introduction

The goal of our project was to build upon the already-implemented website processes for an oral presentation evaluation website for the Colorado School of Mines. It is to be used throughout all of Mines' courses/projects, serving as a more viable alternative to current services such as Google Forms or CATME. Google Forms, though intuitive and inexpensive, has the major issue of being heavily time-consuming, along with not being compliant with the Family Educational Rights and Privacy Act, or FERPA. FERPA specifies that any feedback data among students should not be viewed by any uninvolved third parties, which Google Forms does not abide by. Similarly, CATME presents two major issues. Firstly, if a student makes an inappropriate or offensive comment, the instructor is not able to withhold that singular comment and instead must block all comments from the class, preventing other constructive feedback from ever reaching the students. Secondly, CATME is an expensive service that does not allow for the tracking of usage by department. Being able to keep track of the survey/user count by the department would help with proper cost distribution. We were tasked to finish the functionality of this website. Some of the requirements for this were a page where the instructor can view all comments (sorted by who made the comment to whom/what group) with the ability to withhold singular comments, a function to tally up and average a person's score, and a CSV file-export function that allows an instructor to input grades into Canvas with ease. Additionally, the current program did not contain a view for department administrators to edit/manage the list of superusers for that department – a superuser defined as either an instructor/professor or an authorized TA. Lastly, the superuser should have the option to release comments anonymously or with personal identifiers (first name/last initial).

II. Functional Requirements

This project took the previous EDNS Oral Presentations project and continued its development to add functionality and features. As far as additions to the current code base, we added the front-end portion of the feature for administrators to keep track of the number of surveys taken in their department. This is so that the billed admin can charge each department based on usage of the site. Further development is needed on the back-end/database side. Another feature that was added is the ability for superusers to view all comments given to and by each student on an intuitive page that minimizes the need to click through several times. This page also allows the superuser to redact single comments that may be inappropriate to keep them from being released to the student while allowing all other comments to still be released and viewed. Along with this, the superuser can choose whether or not the comments are released to students anonymously. The website currently has the ability for the instructor to export a CSV file containing all the data for each student's grade, comments, etc… to allow it to be easily viewed and transferred to canvas. This CSV file was modified to fit closer to the client's needs, though future editing must be done to include the history of comments for any one survey.

The complexity of the current existing code base made it necessary to update/further detail the current documentation of the project. Though the previous documentation proved extremely useful when starting out fresh, it lacks the level of specificity needed for a person with no prior knowledge to be able to excel at it. Therefore, we reviewed all of the documentation left by the previous group and edited it to make it easier for the next group to begin working on the project more efficiently. We also communicated with our client expressing willingness to meet with the incoming group to make it as smooth of a transition as possible.

III. Non-Functional Requirements

In order to successfully implement our functional requirements into the end product, we considered how we wanted the system to behave and what limits we needed on its functionality. Our non-functional requirements focused on the behavior, features, and general characteristics of the system's functions. The end goal of our project was to create a website that is compliant with the Family Educational Rights and Privacy Act (FERPA). This is a federal law that protects the privacy of students and education records. This required the system to manage the information each user has access to. Full functionality also required the system to authorize different users and correctly determine different roles and which administrative functions (if any) they could perform. We implemented different roles with different levels of authorization: administrator, superuser (also called instructor), and student. Each department has one administrator (department admin) and super-users can be instructors or TAs of classes within that department. The department admin sets the categories and types of questions superusers can add to their surveys. They also decide the scale of grading criteria (e.g. pass/fail, excellent/needs improvement/poor) and their respective weights (e.g. 100% / 85% / 70% / 50%). These determine the rating values of the Harvey balls used in the surveys under that department. Additionally, the department admin has the ability to add superuser accounts and, in future work, will have access to a count of total surveys taken in each department for billing purposes. Superusers are able to create surveys, set start/end times and dates, export data to a CSV for grades and records, and review, withhold, and release comments.

Another non-functional requirement was to make the website easy to use and enhance the user's experience. The website has the general characteristics that met user requirements and increased usability. Specifically, the buttons include both text and color and the website keeps a consistent color scheme. Both of these additions improved readability. Furthermore, we edited the website to allow superusers to review and withhold comments on a single page. We also condensed the data displayed on this page so they are only viewing the necessary information needed for the purposes of withholding and releasing comments and grading. Lastly, to improve the flow of the website for students, we modified a page that gives students easier access to review the results of their surveys.

IV. Risks

Some risks of this project include maintenance and control of the website. The website also needed to fulfill all the needs of the professor regarding readability and understanding of the data. This also means that the professor needs to understand how to use the website to the best of its capabilities and for their needs. Only one team member had experience with SQL databases, which required the rest of the team to learn the program and language to use with the website. The current plan for who maintains the website and the admin lists are the department heads, which will be considered admins themselves. The admins will be the ones to add or remove admins, which means they will be controlling the website together. Admins will need to know who and how to add to the admin. list and how to add professors as superusers.

V. Definition of Done

The website is operational and usable (on a locally hosted level) by any Mines professors, department heads, and students efficiently. The data from the surveys is presented in a precise and useful manner. A CSV file contains all useful information about each student as well as all comments made and if they were retracted for records. Comments can be held back individually and the rest of the comments can be released to the rest of the students. The data that is taken from the survey is readable on a large scale – the professor can scroll through all comments and decide to withhold a specific comment without issues. The previous website was modified to be more user-friendly and intuitive, which included renaming some titles and adding mouse listeners to increase usability. Finally, a lot of testing was done to make sure the front-end has no errors, the backend has no errors, the API correctly moves data, and all action requests are properly executed.

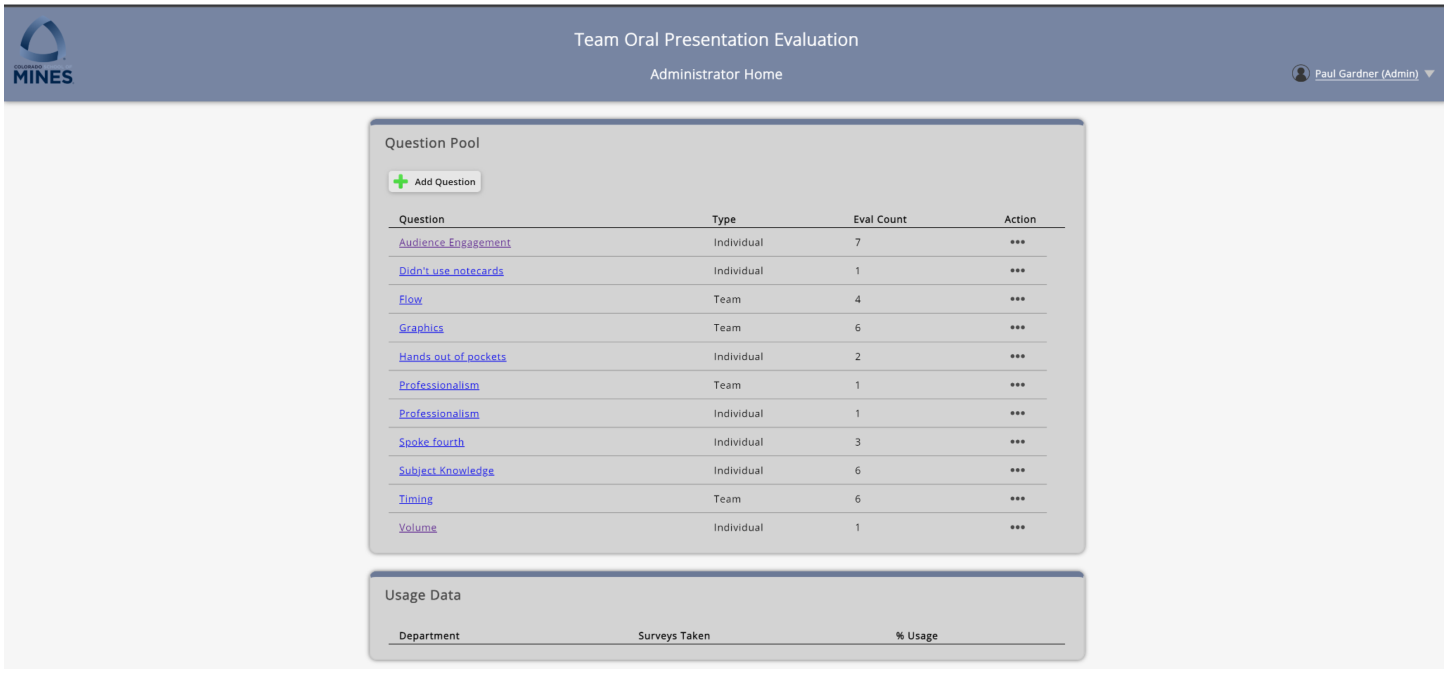
VI. System Architecture

This application has a three-tiered system where each tier has different roles and permissions. The top role is that of Administrator, below that is a Superuser (from now on Superuser will be referred to as Instructor), and finally the Student role. Administrators have the most permissions out of all the roles and have the ability to add/create Instructor users, among many other actions. Instructors hold the ability to create evaluations from the questions provided by the Admin. Students hold minimal permission as their main role is to take the evaluations assigned by the Instructor.

Administrator

When an administrator logs into the website they are brought to the administrator home page, where they will have the ability to do three things: create/modify the question pool from which Instructors will create surveys, create/modify user authorizations for new Admins or Instructors, and adjust the weights for each level of Harvey Balls to be used in surveys.

Admin Home Page View



Team Oral Presentation Evaluation

Administrator Home

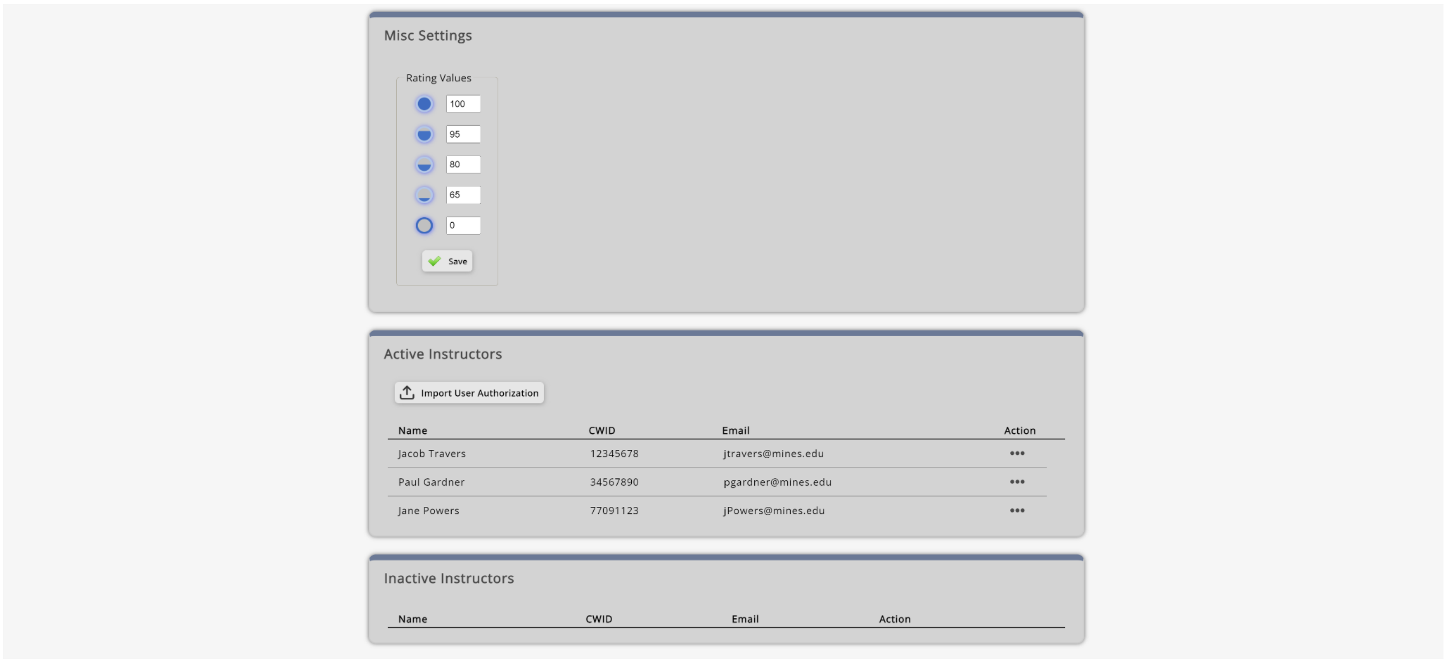
Question Pool

Question	Type	Eval Count	Action
Audience Engagement	Individual	7	•••
Didn't use notecards	Individual	1	•••
Flow	Team	4	•••
Graphics	Team	6	•••
Hands out of pockets	Individual	2	•••
Professionalism	Team	1	•••
Professionalism	Individual	1	•••
Spoke fourth	Individual	3	•••
Subject Knowledge	Individual	6	•••
Timing	Team	6	•••
Volume	Individual	1	•••

Usage Data

Department	Surveys Taken	% Usage

Admin Home Page View (cont.)


Misc Settings

Rating Values

●	100
◕	95
◑	80
◔	65
○	0

Active Instructors

Name	CWID	Email	Action
Jacob Travers	12345678	jtravers@mines.edu	•••
Paul Gardner	34567890	pgardner@mines.edu	•••
Jane Powers	77091123	jPowers@mines.edu	•••

Inactive Instructors

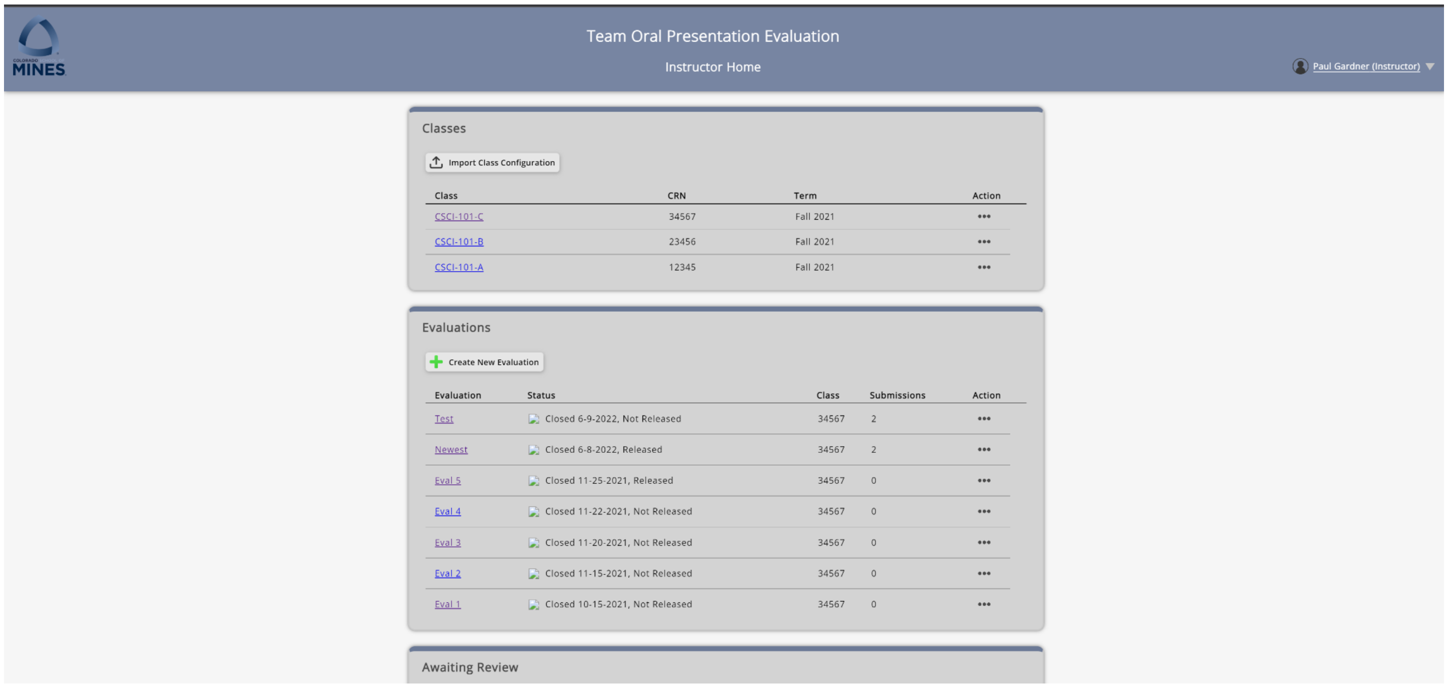
Name	CWID	Email	Action

Per request from our client, all Admin will have the authorization to act as a "Global Administrator" as opposed to having one Global Admin. Additionally, an admin, once it is implemented by the incoming team, will be able to view department usage data. This feature will keep track of survey/submission count and percentage of use based on the total time that the website is hosted on AWS per billing period, to allow for departments to be billed accordingly.

Instructor

When an Instructor logs in, they are brought to the Instructor's home page which currently holds three sections. The first section shows the list of classes that the Instructor oversees and has the ability for them to import new classes via a CSV file containing all information for the class and its students. The instructor can manually add/remove students from a class from here. The next section shows all evaluations created by the instructor and all pertinent information such as open/close date, CRN, and the number of survey submissions. Each evaluation can be modified until after the close date. The instructor can also create new surveys using the pool of questions provided by their department administrator, as well as select questions for the team presenting or individual students.

Instructor Home Page View



Team Oral Presentation Evaluation

Instructor Home

Paul Gardner (Instructor)

Classes

Import Class Configuration

Class	CRN	Term	Action
CSCI-101-C	34567	Fall 2021	•••
CSCI-101-B	23456	Fall 2021	•••
CSCI-101-A	12345	Fall 2021	•••

Evaluations

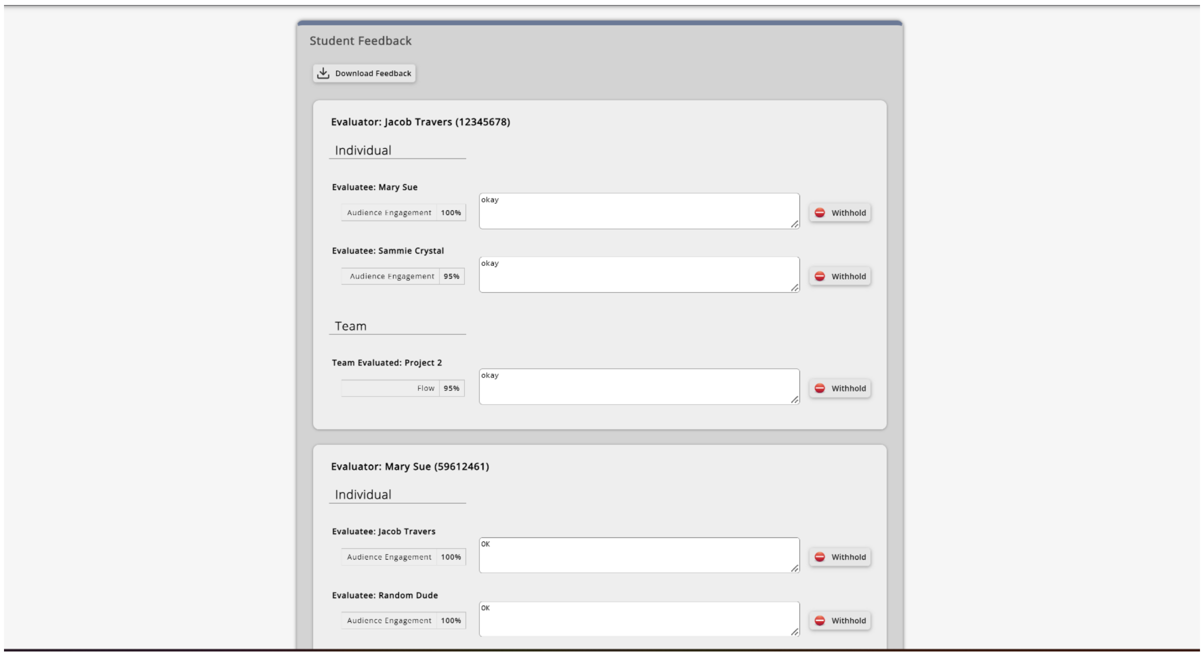
Create New Evaluation

Evaluation	Status	Class	Submissions	Action
Test	Closed 6-9-2022, Not Released	34567	2	•••
Newest	Closed 6-8-2022, Released	34567	2	•••
Eval 5	Closed 11-25-2021, Released	34567	0	•••
Eval 4	Closed 11-22-2021, Not Released	34567	0	•••
Eval 3	Closed 11-20-2021, Not Released	34567	0	•••
Eval 2	Closed 11-15-2021, Not Released	34567	0	•••
Eval 1	Closed 10-15-2021, Not Released	34567	0	•••

Awaiting Review

The final section the instructor has access to is the awaiting review section. This is where the instructor can view the results of the surveys and all of the comments given to/from students. From here the instructor can choose to redact certain comments if they are too vulgar or inappropriate to be released. There is also the option for the instructor to choose to release the comments anonymously or not depending on the class.

Instructor feedback view



Student Feedback

Download Feedback

Evaluator: Jacob Travers (12345678)

Individual

Evaluatee: Mary Sue

Audience Engagement 100%

okay

Withhold

Evaluatee: Sammie Crystal

Audience Engagement 95%

okay

Withhold

Team

Team Evaluated: Project 2

Flow 95%

okay

Withhold

Evaluator: Mary Sue (59612461)

Individual

Evaluatee: Jacob Travers

Audience Engagement 100%

OK

Withhold

Evaluatee: Random Dude

Audience Engagement 100%

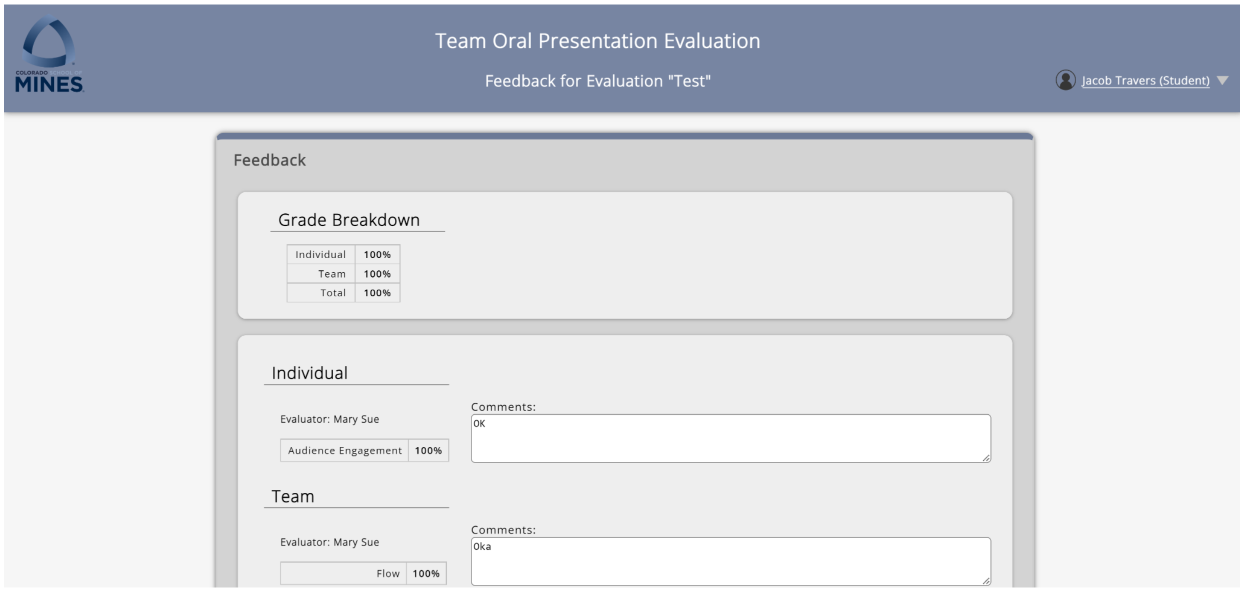
OK

Withhold

Student

When a student logs in, they are brought to the student home page. There are four sections shown on the home page. The first section contains the list of evaluations that they need to take. The student clicks on the evaluation that they need to take, and it brings them into the evaluation. Here, they select which team they are evaluating, and the evaluation is populated with the students that are on that team. The student then selects which Harvey Ball rating they feel the student/team deserved. The ratings for each ball are displayed at the top of the evaluation to ensure that the student indubitably knows which grade they are giving. At the bottom of each section, there are boxes to provide a comment for each student. These are set to be required or optional by the instructor that created the survey, or any of the TAs who may modify it afterward. When they have filled out the entire evaluation, they can submit the feedback. If they do not correctly fill out the form, they are given an error and the form is not submitted. The next section shows the upcoming evaluations that they eventually need to take. These are then moved into the first section when they open. The third section contains past evaluations that they have taken for other students. The evaluation moves to this section once it has closed. The fourth section contains the feedback from other students that the student received. It has a breakdown of their individual score and comments along with their team score and comments. They are only able to see their team score, and their personalized comments. This calculates the total score they received on that presentation based on how the professor set the weight for each section.

Student Feedback View: Evaluator Released



Team Oral Presentation Evaluation

Feedback for Evaluation "Test"

Feedback

Grade Breakdown

Individual	100%
Team	100%
Total	100%

Individual

Evaluator: Mary Sue

Audience Engagement	100%

Comments:
OK

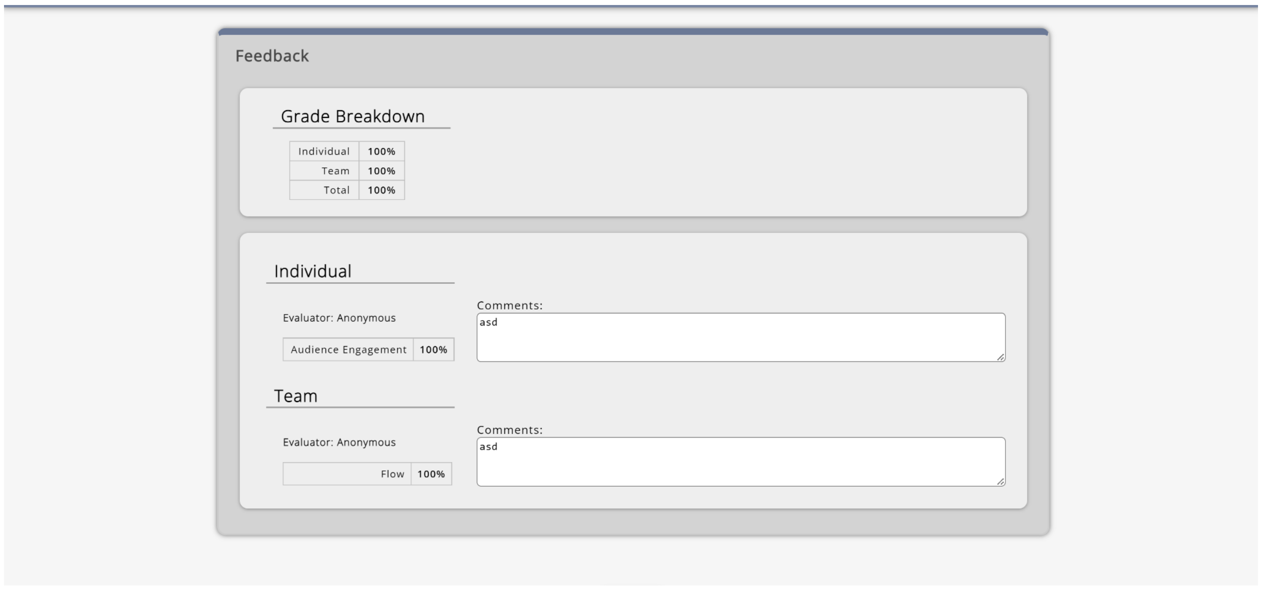
Team

Evaluator: Mary Sue

Flow	100%

Comments:
Oka

Student Feedback View: Evaluator Released Anonymously


Feedback

Grade Breakdown

Individual	100%
Team	100%
Total	100%

Individual

Evaluator: Anonymous

Audience Engagement	100%

Comments:
asd

Team

Evaluator: Anonymous

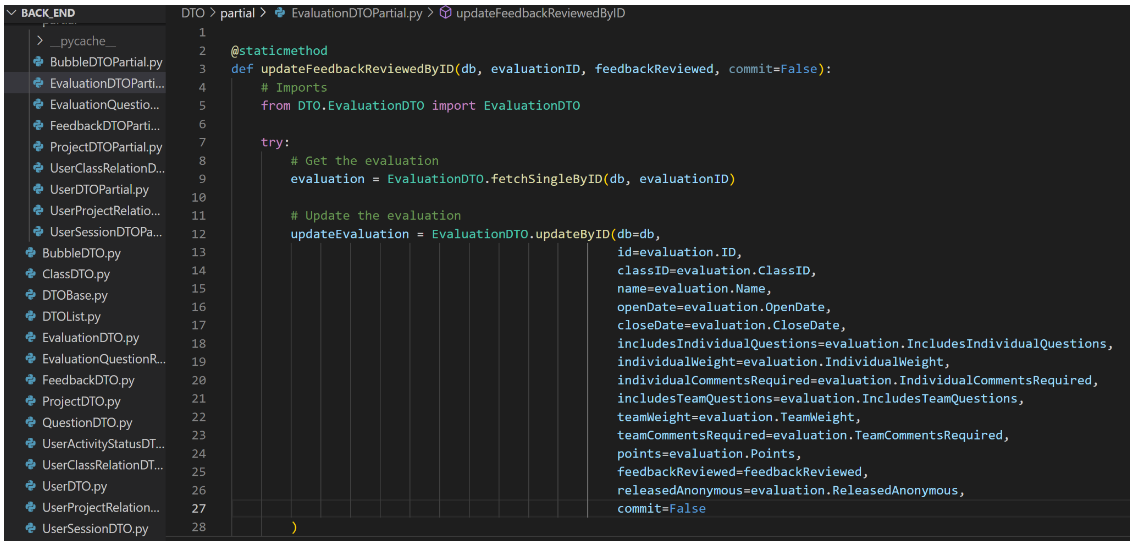
Flow	100%

Comments:
asd

VII. Technical Design

Our website was implemented using JavaScript React on the front end and Python with a Flask API on the back end. We used MySQL for our database and localhost for our server. The purpose of the back end allows communication between the front end and the database. In order to optimize this communication, our back-end code uses several auto-generated Data Transfer Objects (DTOs) as well as several hand-written partial DTOs. This enabled us to send very specific requests to the database whenever necessary rather than having to write specific SQL requests each time information from the database is needed. There is a DTO for each table in the database. For example, table User_T has a file UserDTO.py that creates the UserDTO class. Each DTO class handles SQL queries for fetching, updating, inserting and deleting data along with a function that attaches custom data to a DTO object. The figure Back-End Code DTO Implementation below shows the definition of UpdateFeedbackReiviewByID for EvaluationDTO.py

Back-End Code DTO Implementation



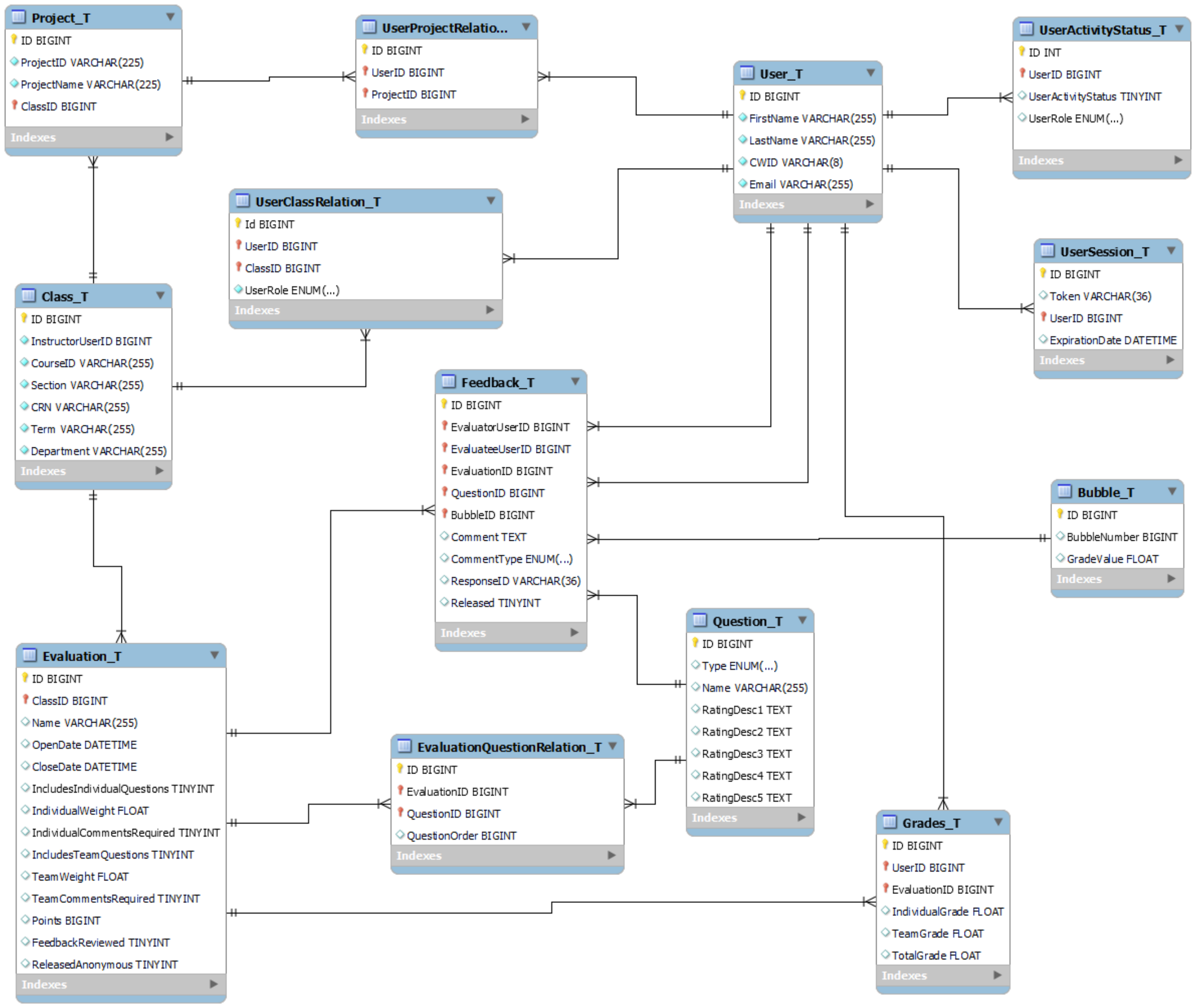
```python

@staticmethod
def updateFeedbackReviewedByID(db, evaluationID, feedbackReviewed, commit=False):
    # Imports
    from DTO.EvaluationDTO import EvaluationDTO

    try:
        # Get the evaluation
        evaluation = EvaluationDTO.fetchSingleByID(db, evaluationID)

        # Update the evaluation
        updateEvaluation = EvaluationDTO.updateByID(db=db,
                                                    id=evaluation.ID,
                                                    classID=evaluation.ClassID,
                                                    name=evaluation.Name,
                                                    openDate=evaluation.OpenDate,
                                                    closeDate=evaluation.CloseDate,
                                                    includesIndividualQuestions=evaluation.IncludesIndividualQuestions,
                                                    individualWeight=evaluation.IndividualWeight,
                                                    individualCommentsRequired=evaluation.IndividualCommentsRequired,
                                                    includesTeamQuestions=evaluation.IncludesTeamQuestions,
                                                    teamWeight=evaluation.TeamWeight,
                                                    teamCommentsRequired=evaluation.TeamCommentsRequired,
                                                    points=evaluation.Points,
                                                    feedbackReviewed=feedbackReviewed,
                                                    releasedAnonymous=evaluation.ReleasedAnonymous,
                                                    commit=False
        )
```

Database Schema



VIII. Software Test and Quality

Testing Software:

- Postman:

The Postman software allows for testing the communication between the front-end and the database by sending specific queries through the database. This is particularly useful for backend coding/debugging by eliminating the need to have the website and front-end code running while still being able to send requests through the back-end to the database and ensure proper security/privacy and properly formatted information.

- Google Chrome in-browser debugger:

The chrome extension added for this project allows us to have both the website and front-end code pulled up in the same window. This means that we can quickly make small edits and changes to the code and instantaneously reload the page to see how these changes affect the website. This helps us be able to quickly revert code back in case the modifications negatively affect the website.

- Website:

Since a lot of the functionality requirements and changes we will be making directly affect the usability of the website a lot of our software quality tests will simply be loading up the website after making changes to the codebase. There we can then step through the appropriate pages and buttons in order to assure the changes made are functioning as intended without breaking other aspects of the site.

- Using active/inactive variables on all questions and users:

Giving each user and question a variable determining if they are active or inactive makes it so that even once “deleted” through the website they remain in the database with the “inactive” setting. This helps assure that errors stemming from deleted users or questions could easily be reversed as well as giving Administrators the ability to look through the database and still see past questions and users.

- Code Reviews:

After finishing changes to the code, the written code is reviewed and tested through the website by another programmer to ensure no errors slip through to the final push of the code.

Testing Results:

Heavy user testing was done on several different machines to make sure that the locally-hosted website worked properly in different settings. The current program set-up works on Windows 10/11, as we ran into issues installing necessary programs (MySQL in particular) on both a Linux machine and MacOS. Based on the results of our test, error handling has been done exceptionally well and the website runs smoothly. Users are able to be added/removed from a class, as well as take/review feedback surveys. Additionally, instructors (superusers) are able to view all feedback given across the class, along with what student made what comment to whom/what team, with identifiers of First/Last name. On the administrator side of things, instructors can be added/removed to/from a class, and all instructors in that department can be seen from the administrator home page. Testing for these user-related features was done through the usage of MySQL, a Flask server API for the backend (using Python), and JavaScript React for the front-end.

IX. Project Ethical Considerations

The project was created with the Mines campus in mind. It was meant to benefit the department heads, professors, and students. As we developed our project, we must also consider whether we were adhering to the ACM/IEEE ethical expectations for software engineers. For our specific project, there were four pertinent principles: Client and Employer, Product, Management, and Self.

I. Client and Employer - Software engineers shall act in a manner that is in the best interests of their client

   We made sure to communicate with our client any limitations that made a feature out of our scope to implement (Principle 2.01 and 2.06)
   - 2.01: Let our client know what a realistic timeline is for completion of certain features.
   - 2.06: Kept our client updated on the progress made on the project in case the client wanted us to go a different route.

II. Product - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible

   We strived for high-quality software on a reasonable schedule (Principle 3.01), set and document achievable requirements and specifications (Principle 3.02, 3.07, and 3.08), performed documentation and testing to ensure software quality (Principle 3.10 and 3.11), and developed secure software with the privacy of users in mind (Principle 3.12 and 3.14).

- 3.01: All new features were added using a template to keep the software as high quality as we could.
- 3.02: Everyone created goals that were achievable in a certain timeline and didn't overload themselves.
- 3.07: We scheduled multiple client meetings a week to keep the client updated on our work. We also made sure that the features that were added to the project were correct and the way that the client wanted them to be.
- 3.08: Each change was pushed into the Github repository with a clear description of what the change was and why it was done.
- 3.10: Every time a change was implemented on the project, the modification was tested with general use and edge cases. Also, with every change, each group member had a chance to review the code and functionality before moving on to the next feature.
- 3.11: Confluence continues to be used to document how the project is put together and how it works. It also covers how to install all the software required to start working on the project and what problems the project has or may have that need to be addressed in future work.
- 3.12: The project was created with privacy in mind. All features have been implemented to make it easier for the instructors to keep information private and released to the correct people.
- 3.14: All data that is being used for testing is dummy data, it isn't tied to any real person and is solely used for testing current code to make sure real code doesn't appear where it doesn't belong.

III. Management - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance

We understood the clients' needs for protecting sensitive information (Principle 5.03), including but not limited to grades and login information such as usernames and passwords.

IV. Self - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

We must further our knowledge, ability, and understanding of developing safe, reliable software (Principle 8.01 and 8.02), creating solid documentation (Principle 8.03), the environments and documents which we will be working with (Principle 8.04), the standards and laws related (Principle 8.05), and the ethical principles themselves (Principle 8.06).

- 8.01: Engaged in an agile development method to tie all the code together and prevent anything hardcoded.
- 8.02: Utilized a template for all functions that checks authorization for every request.
- 8.03: Used Github to track all features added and a description of the feature. Confluence documentation for descriptions of components and how they work together.
- 8.04: Scheduled meetings with a member of the previous group to explain how the code worked in detail and how to edit it for the new features that we implemented.
- 8.05: Followed templates for how to name variables and create functions. Included in the template, security checks for keeping unauthorized users from accessing private data.
- 8.06: Having no prior knowledge of the code, learning the new syntax and functionality of the coding languages was a must.

Though we did not deal with concerns regarding the unethical use of this software (with a loose definition of “being used for evil”), we had to ensure the security and clean protocol execution of the website, so no skilled unauthorized user is able to freely do with the website as they wish. The concerns of an authorized user abusing the privilege they are given are handled by the contracts staff have to sign when entering a new position at Mines.

X. Project Completion Status

We were the second group to work on this project. We were handed down a functional website that had flow and components already created with some features already implemented. Most of our work surrounded implementing new features and fixing bugs along the way.

Main features that were implemented:

- Active/Inactive Users
  - Each user now has an activity status that can be turned active/inactive to keep records but add/remove access
- Editing Instructors
  - Admins now have new sections on their homepage to edit instructors
  - Editing instructors includes editing their name, email, CWID, and what classes they have access to
  - Activate or deactivate an instructor
- Anonymous feature for returning feedback to students after it has been released.
  - Option to display the name of who sent the comment to the evaluatee
- Added the evaluator name next to feedback for the instructor to see when reviewing feedback
  - Both names are included which gives the instructor the information of who said what to who.
- Kept all backend checks with authorization and errors with each feature that was implemented
  - If a user somehow got to the backend side past the front-end redirection, the backend still verifies that a user has permission to use that function.
- Fixed a bug for displaying classes that an instructor is included in
  - If a student is made an instructor (TA), it would send all the classes that they are a student in as well. Implemented a checker to make sure it only sends classes that they are an instructor for.
  -

XI. Future Work

Due to initial time constraints regarding the proper installation and set-up of the development environment, along with difficulties parsing through existing code, not all of the intended features were implemented. In particular, we detail below features that either were discussed with the client but were outside of our scope or ones that we ran into issues with and could not complete.

- There is not currently a proper login page set-up. Due to this project being hosted locally and not on an AWS server, creating a drop-down menu with the different authorized/unauthorized users that were populated from the database directly made the most sense.
- The administrator of a department should not worry about class assignments. The current set-up of the webpage makes it so that the administrator must manually edit each and every instructor/TA and assign them (or remove them) from a class. This is a considerable amount of work put on the administrator that should be fixed for a better alternative at a later date.
- Administrators need to see the campus-wide by-department usage of the website. There is currently progress done on the front-end portion of the website towards the completion of this feature, but the current back-end implementation needs work.
- Instructors must be able to take their own surveys. The implementation of this feature is so that instructors/TAs of a class are able to evaluate students presenting, in a similar manner to what is done currently in the CS@Mines Field Session, CSCI370.
- Instructors should be able to create classes and add other admin-permitted instructors to their classes. This point ties to the previous one, in that, by removing the necessity of an administrator to manually assign all instructors/superusers to classes, we must implement an alternative. With this, an authorized instructor is able to freely create and modify the parameters of a class, as well as its members/other instructors.
- Questions from the question pool on the administrator home page should not be removed, but rather placed in an inactive/active status section. This implementation is particularly similar to the current (implemented) feature

of inactive/active instructors. In particular, a question should only be able to be placed in the inactive portion if it is currently not being used in a survey. It is already present within the database but needs to be added as an actual feature to the website.

- Within the grade-populated CSV on the instructor view of a survey, there should be a time stamp on each person's submission, so as to be able to sort through legitimate/tardy submissions for the survey in question. Additionally, there is an error where the survey deadline is 6 hours off its actual closing time; however, this latter issue was not addressed due to concerns with porting to AWS and how that might affect our potential, hard-coded solution.
- Administrators of a department should be able to add other administrators to other departments. In particular, this addresses the issue of "What happens if the sole administrator of a department loses access to their account?". This issue should be handled only once the website has been successfully ported to AWS.
- When an instructor is viewing all feedback given for a particular survey, they should be able to sort the feedback given by teams. This will make it easier for multiple-instructor classes to be able to grade their assigned teams without the clutter of other teams.
- Whenever a survey is created by an instructor, they should have the option to keep the survey at hand private (limited to their account), or pass it onto the administrator for further review and the potential of making it public (where all instructors are able to utilize this survey "template"). This is so that instructors do not have to manually create extremely similar surveys from scratch over and over.
- The website needs to be ported to AWS. Though initially, this was one of our goals for the field session, due to time constraints and issues acquiring a Mines-affiliated AWS account this quickly became out of our scope. Future students who invest time on this project should have this as their primary focus at the start, as this will potentially create many issues that will need to be solved in a timely manner.

Finally, some of the skills that the next team to work on this project might benefit from are the following:

- Database Management/MySQL
- JavaScript React for the front-end
- Python, Flask server API for the back-end

XII. Lessons Learned

Here are some lessons we learned throughout our project and its testing procedures:

- Though our initial intent with the project was to divide up the work by areas of expertise (i.e. a few of us working on the back-end, some of the front-end, etc.), we quickly learned that this process would not set us up for success. This is because features that we needed to implement required, for the most part, interaction from all three portions of the project (database - back-end - front-end). It was a lot simpler to take a feature through all of the processes necessary.
- It became increasingly obvious that in order to be able to succeed at this field session, we had to appropriately split up work. Had we not taken the time to discuss what needed to be done for the week along with who was accomplishing what task and what help they might need, we would have come crashing down at an accelerated rate.
- The Google Chrome Developer Tools proved to be extremely useful when it came to minor editing of the website. In particular, items such as changing all instances of "superuser" to "instructor" for easier accessibility, and modification of the order of items displayed on the instructor page, among others.
- Reusability of code was key in the implementation of many new features. In particular, the code for a checkbox for instructors to be able to release feedback anonymously (or not), which was also used when the admin is to choose what instructors are tied to what classes, came from the checkbox feature for instructors to make a survey individual/team-based.
- Being able to have administrator access to the machine where tests were run was necessary throughout this project. There were a lot of co-dependencies when first installing the necessary programs that required the

setting of user permissions for scripts and other things which would have not been possible on a campus-hosted machine.
- Meeting with the client and showing the working product, making the edits they want to be implemented, and repeating this process was the most effective way we found of making a product the client was satisfied with. This matches up with the recommended development process in the industry, where client consultation is regularly done.

XIII. Team Profile

Gabriel Del Castillo:

My name is Gabriel Del Castillo. I am an incoming Junior in Computer Science with a focus on Robotics and Intelligent Systems. I spend my free time hanging out with friends, playing video games, and looking for opportunities to get ahead in my career.

Thomas Lowery:

I am Thomas Lowery, about to be a Senior in Computer Science with a focus in Computer Engineering. I enjoy breakfast food and weightlifting.

Michael Kormishkin:

My name is Michael Kormishkin. I am a Junior in Computer Science on the general track. I am planning on obtaining a Cyber Security certificate and an unofficial emphasis on Business. My hobbies include mountain biking, hiking, driving, and some video games here and there.

Alexandra Cooke:

My name is Lexi Cooke and I am going to be a Senior in Computer Science this Fall. I enjoy skiing, boating, and my dogs June and Mochi.

Appendix A – Key Terms

Term	**Definition**
API - Application Programming Interface	*Tools and resources that allow developers to create software applications (websites, apps, etc.)*
AWS - Amazon Web Services	*An Amazon subsidiary that provides on-demand cloud computing platforms and APIs*
Family Educational Rights and Privacy Act (FERPA)	*This is a federal law that protects the privacy of students and education records*
DTO - Data Transfer Object	*Carries data between processes and facilitates communication between the API and the server*