Ramblin-Wreck Review

CSM EDNS 3 - Team Evaluation

Brody Clark, Jarod Clark, Nathan Mackey
Donna Bodeau

06/14/2021

Ramblin-Wreck Review
EXPLORE YOUR TEAM

Table of Contents

Introduction

Client Description

Professor Donna Bodeua has a B.S. in Electrical Engineering from Gonzaga University. She currently teaches Intro to Design and Practice of Design at the Colorado School of Mines. On top of that, she is an advisor for Advanced Software Engineering at the Colorado School of Mines.

Project Description

The Colorado School of Mines currently contracts Comprehensive Assessment of Team Member Effectiveness (CATME) as a peer evaluation tool for group projects in the Engineering Design and Society (EDNS) department. Professor Donna Bodeau proposed a new peer evaluation platform run through Mines to avoid additional costs of CATME contracts, provide a customizable survey for all departments, and allow for better functionality that CATME doesn't have. CATME currently costs a small university, such as Mines, $2 per student to use their software which can get expensive when you have a couple thousands kids using this software every year. Aside from cost, CATME allows a professor to create a peer evaluation survey with a multiple choice performance poll and a free response self/peer evaluation questionnaire. The multiple choice questions are chosen by the survey's creator on setup from a provided subset of categories to judge a peer's performance. Although they get to choose what their survey includes, they only choose topics which don't allow for a user to preview this survey and actually know what the students will be asked. During setup, the professor chooses to release comments made during the free response to students and whether or not the feedback will be anonymous. The problem with this functionality is CATME doesn't allow a superuser to disable inappropriate comments, once one is disabled they all become disabled. Once the surveys are released to the students, students can take the survey with a one time submission. Only being able to take this survey with a one time submission can be a problem since a mindset about a teammate can change over a couple of days. The answers provided are then transferred to a difficult to read comma-separated values (CSV) file to which a Mines professor must interpret the data and calculate a score for each student manually.

The proposed project will create a new survey platform that is fully customizable and accessible to all departments for no cost to the Colorado School of Mines. The goal of the survey is to be able to access a peer evaluation through Mines multi-pass credentials and allow for authorized users to customize a peer evaluation tailored to their department. Although the software doesn't have this integration quite yet, it is something that will be included at some point. This software contains a back-end database holding information to create a survey.

Administrators are assigned to a department where they have the ability to populate, edit, and remove data. The administrator can create category descriptions which contain questions. Each question is populated with grading descriptions based on that department's delinations. Administrators also have the ability to make a question mandatory throughout the department. As a superuser, they can upload a new class roster which will assign students to their teams and teams to their class. To create their class survey the superuser has to choose a title, start and end date, and a class to send the survey. The survey is populated with the mandatory questions and any questions the superuser selects from the database that the administrator created. Once the surveys are completed, the students in the selected class will start the survey on the start date and have until the day after the end date to edit the survey. Once the end date has passed, the results are sent to the superuser in a simple CSV file. The superuser will ensure any inappropriate comments are removed and then send the results back to the students.

Requirements

Functional Requirements

The main functionalities of the project are to allow an administrator to create the overall category descriptions and corresponding questions for the department, then allow superusers to select which questions they would like for their class survey, and finally allow the students to take and submit the surveys.

- Log in Page that allows a user to login with their correct credentials, and then directs them to the correct page based on their access
    - **Admin Page**
        - Edit the category delineations
        - Insert, edit, and remove assessment categories
        - Insert, edit, and remove questions that correspond to a specific assessment category
        - Allowed to give users their permissions
        - Act as a superuser
    - **Superuser Page**
        - Insert a CSV file that will be read in as a course roster, creating a class, teams in that class, and students in those teams
        - View class, teams, and students
        - Create survey by selecting questions that the administrator created, entering a title, choosing start and end date, and choosing a class to receive the survey
        - Preview survey before it is sent out
    - **User Page**
        - The student may view a list of past and current surveys
        - Based on the start and end date, a student may view results, or go in to take a current survey
        - Go in and out of a current survey until the end date has passed

Non-Functional Requirements

- Project must compatible with the other Design based projects
    - There are two other EDNS projects that will eventually be integrated with this project
- User friendly screens
    - Less click through buttons on the pages
- Course rosters

    - Import a CSV file that can be read in
    - Creates a class in database
    - Puts team into that class
    - Puts students into teams
- Python and JavaScript programming language
    - React for user interface (UI)
    - Django for database, strong web framework

System Architecture

Frontend Interface

Figure 1 shows the overall interaction between each type of user access and the web application. It starts at a login page, moves into each respective homepage, then the actions from the homepage, to everyone getting the results. The horizontal arrows in row 2 show that a user with administrator permissions is allowed to move from superuser access to admin access, but a user with superuser permissions is only allowed to access pages shown vertically connected.

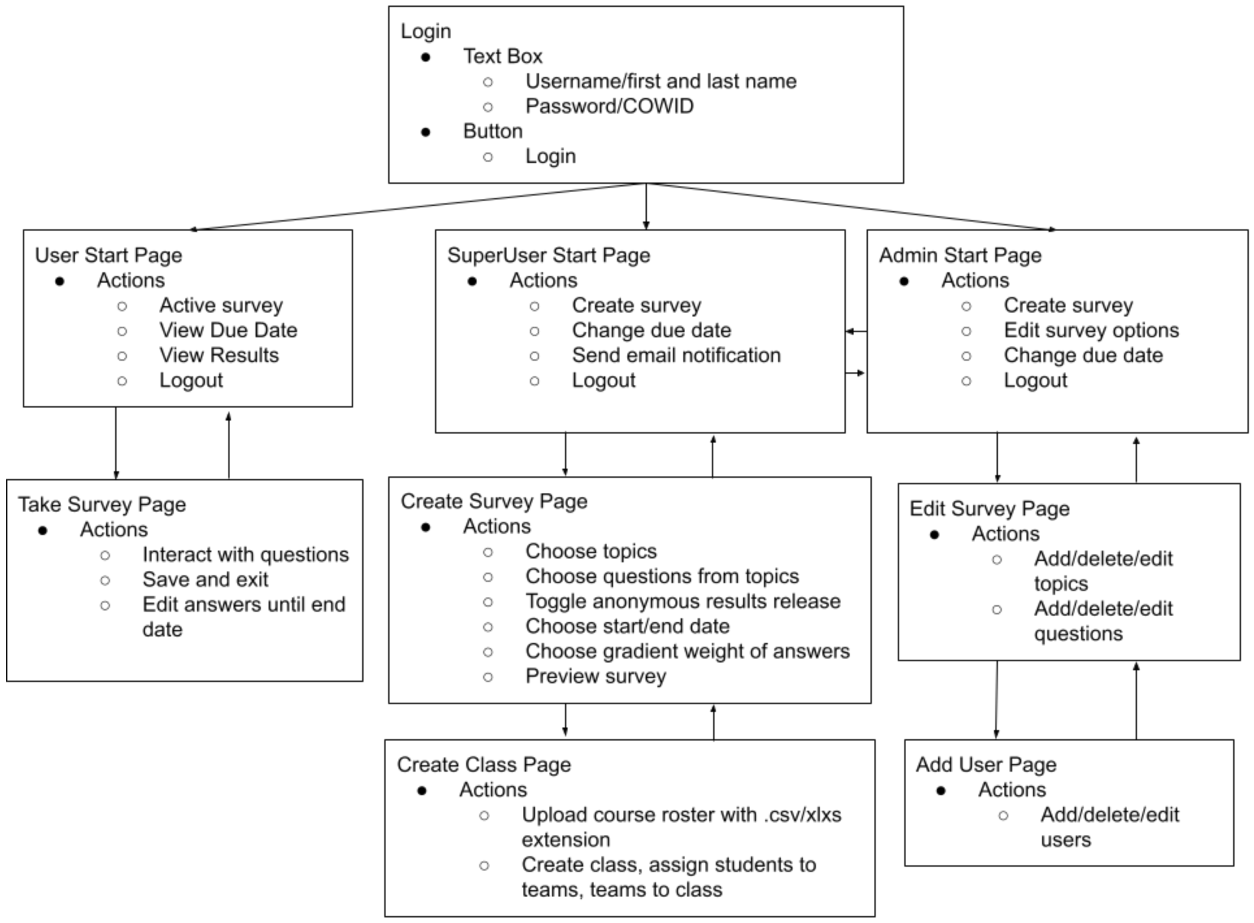
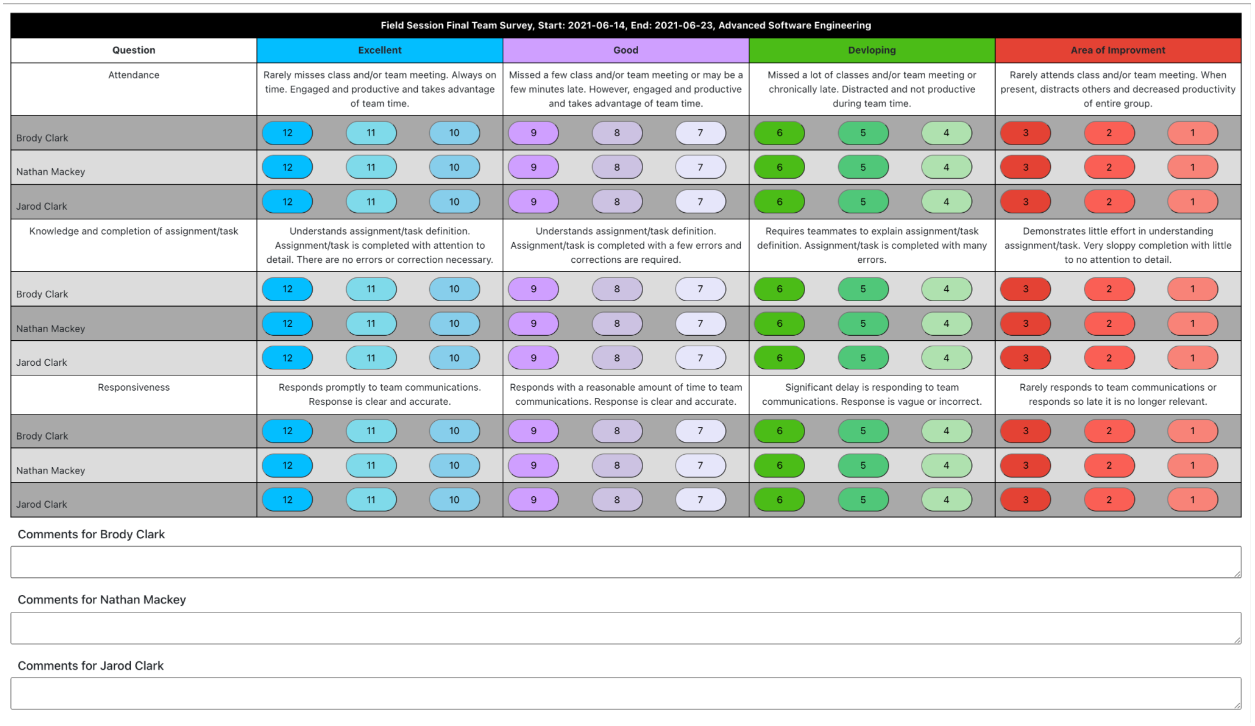


Figure 1: Web Application Flow Chart

Survey Interface

Figure 2 is the prototype of what the survey the user takes currently looks like. The survey shows the question on the left column, categories delineation on the top row, and the category descriptions inside each cell. The circles within each cell allow a click and only one button clicked per row (question). On the bottom of the survey, the text boxes for free response feedback are displayed.



Field Session Final Team Survey, Start: 2021-06-14, End: 2021-06-23, Advanced Software Engineering

Question	Excellent			Good			Devloping			Area of Improvment		
Attendance	Rarely misses class and/or team meeting. Always on time. Engaged and productive and takes advantage of team time.			Missed a few class and/or team meeting or may be a few minutes late. However, engaged and productive and takes advantage of team time.			Missed a lot of classes and/or team meeting or chronically late. Distracted and not productive during team time.			Rarely attends class and/or team meeting. When present, distracts others and decreased productivity of entire group.		
Brody Clark	12	11	10	9	8	7	6	5	4	3	2	1
Nathan Mackey	12	11	10	9	8	7	6	5	4	3	2	1
Jarod Clark	12	11	10	9	8	7	6	5	4	3	2	1
Knowledge and completion of assignment/task	Understands assignment/task definition. Assignment/task is completed with attention to detail. There are no errors or correction necessary.			Understands assignment/task definition. Assignment/task is completed with a few errors and corrections are required.			Requires teammates to explain assignment/task definition. Assignment/task is completed with many errors.			Demonstrates little effort in understanding assignment/task. Very sloppy completion with little to no attention to detail.		
Brody Clark	12	11	10	9	8	7	6	5	4	3	2	1
Nathan Mackey	12	11	10	9	8	7	6	5	4	3	2	1
Jarod Clark	12	11	10	9	8	7	6	5	4	3	2	1
Responsiveness	Responds promptly to team communications. Response is clear and accurate.			Responds with a reasonable amount of time to team communications. Response is clear and accurate.			Significant delay is responding to team communications. Response is vague or incorrect.			Rarely responds to team communications or responds so late it is no longer relevant.		
Brody Clark	12	11	10	9	8	7	6	5	4	3	2	1
Nathan Mackey	12	11	10	9	8	7	6	5	4	3	2	1
Jarod Clark	12	11	10	9	8	7	6	5	4	3	2	1

Comments for Brody Clark

Comments for Nathan Mackey

Comments for Jarod Clark

Figure 2: Survey Preview

Database Schema

Figure 3 shows the back-end of all the data that is currently being used for the website. The database is large and will constantly be changing as the website is used. The admins can change the topics and questions under each department which then changes the allowed choices for the professors which follows what the user sees when taking the survey. The database may change in the future when Mines multiplass is implemented.

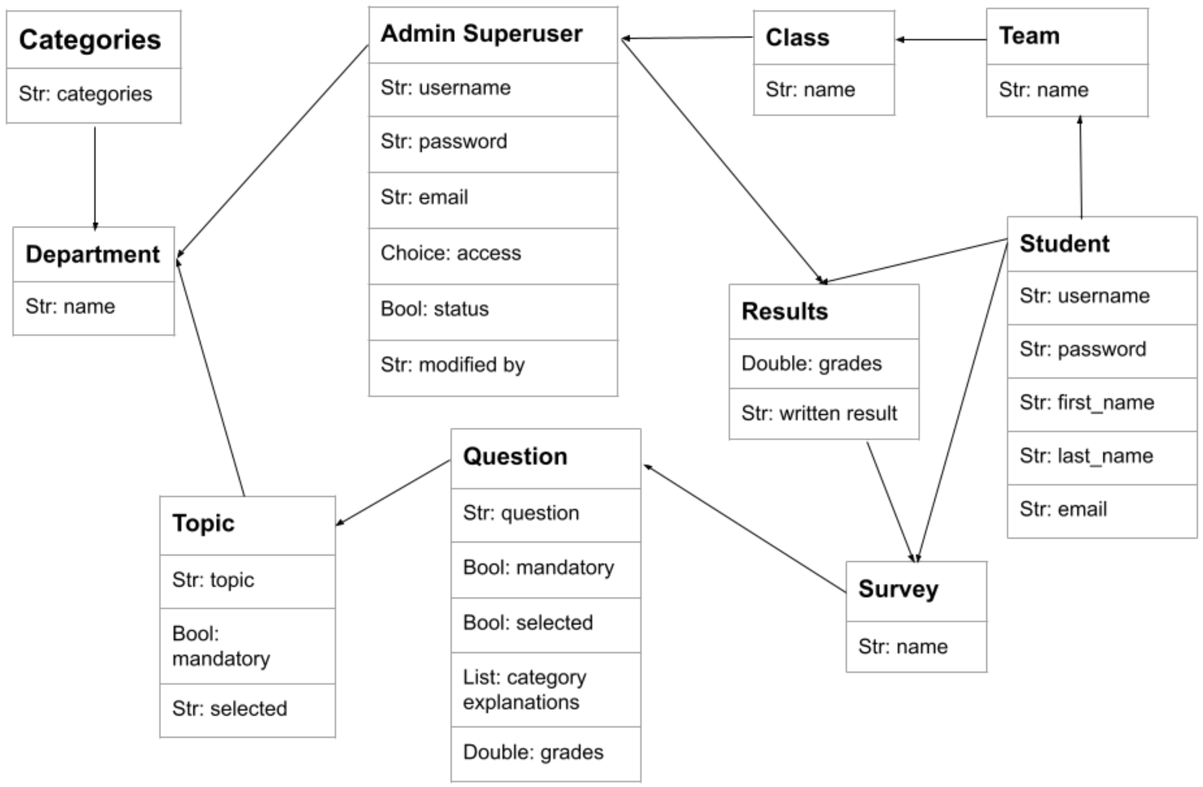


Figure 3: Database Connections

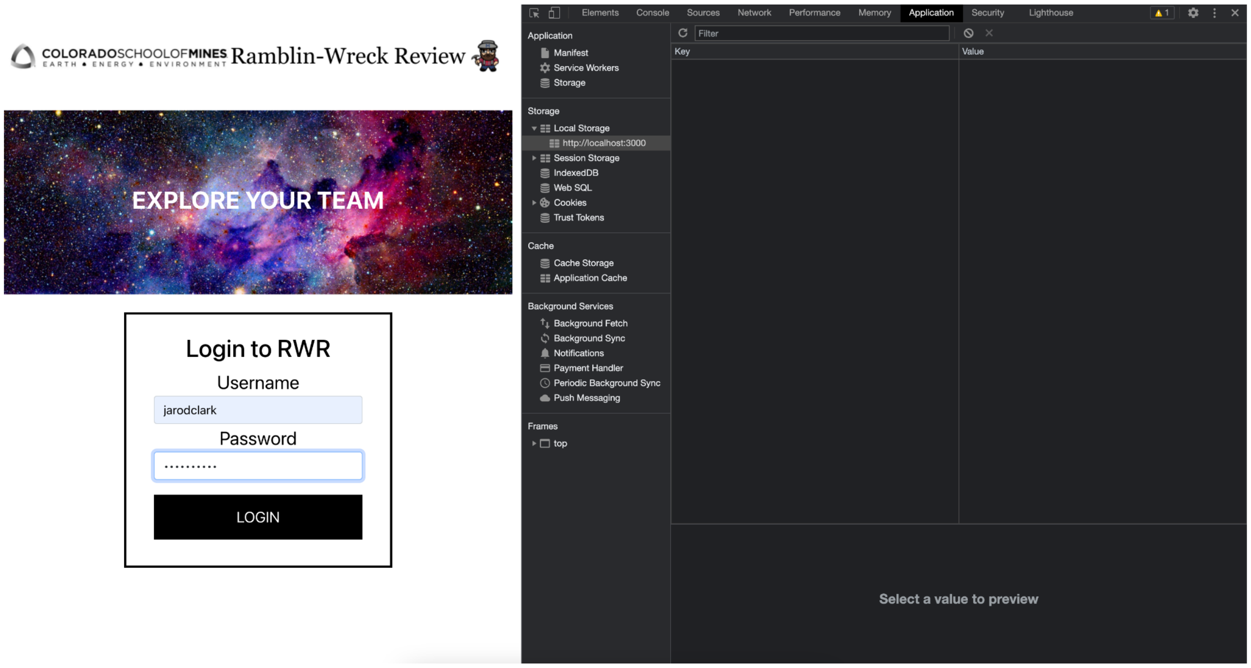
Back-end to Front-end Integration

Django allowed for a complete and neatly designed database server that is easily accessible through the front-end React UI. The software is able to make fetch calls to the database server from the front-end JavaScript program that enables it to get data, post data, update data, and delete data from the database. Being able to have this functionality with the database made it simple for the software to interact with the database as it needed.

Technical Design

Local Storage Utilization

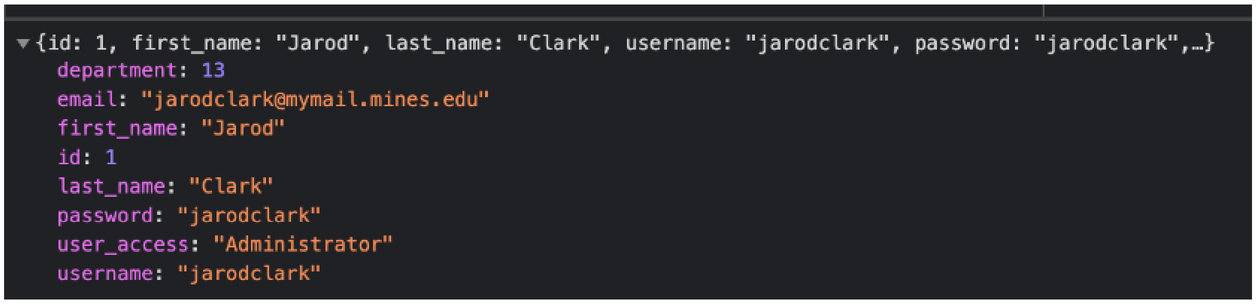
One important aspect of the design is being able to save data and pass data as a user goes from one web page to another or when a page refresh occurs. This problem was faced early on and led to testing and debugging throughout the beginning stages of the software development process. Going from the login page to the next page, the program needs to know who the user is, meaning it needs to transfer the user's data to ensure the program displays the appropriate data. In figure 4, the login page shows the local storage being empty but once the user hits submit with the correct credentials to login, figure 5 then shows the user's data being saved in the local storage. The data shown in figure 5 shows the data in JavaScript Object Notation (JSON) form after it is saved. Once it is saved to the local storage, the program can check to see if data is there, if it is there the software can use it, and if it is not there the software can make sure to save our data to it. Without this process being incorporated into the program, when a page change or a refresh occurs, the page will be blank since there is no data to read from. A way the program does this is by making an export variable from the login page file to the next files. Once it is passed, the user data must be saved to a local storage that is checked and saved to a variable. This is an ongoing process as the program re-renders to ensure that the data is saved, checked, and then set to a variable. All of the other aspects from the website come from the user's data so ensuring the local storage is utilized fixes the problem. The first thing the program does in specific files is either read the local storage and set that data to the user, or write the user's data to the local storage.



Username: jarodclark

Password: ....

Figure 4: Before Local Storage Configured

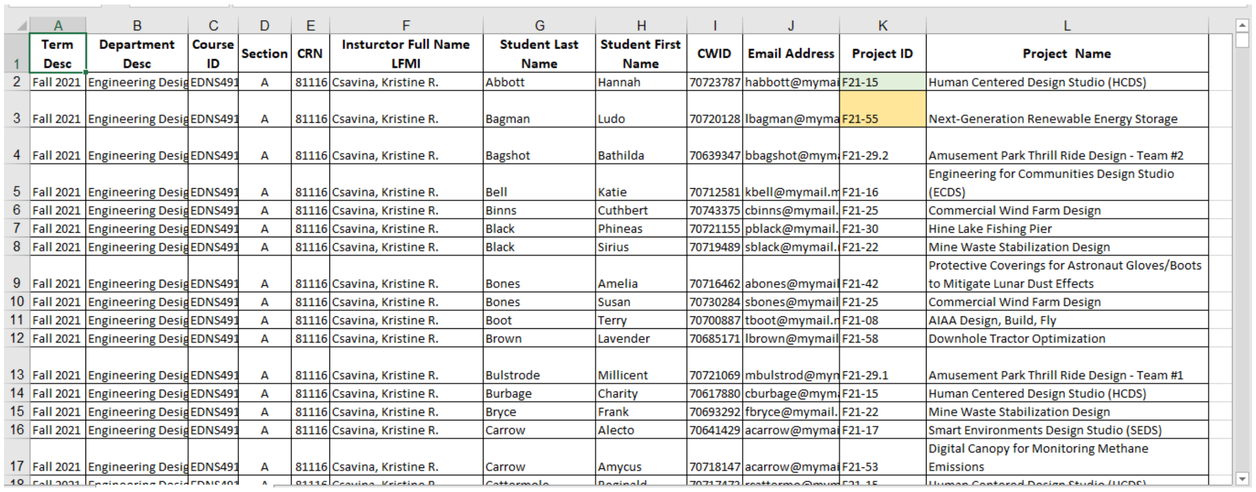


```
▾{id: 1, first_name: "Jarod", last_name: "Clark", username: "jarodclark", password: "jarodclark",…}
   department: 13
   email: "jarodclark@mymail.mines.edu"
   first_name: "Jarod"
   id: 1
   last_name: "Clark"
   password: "jarodclark"
   user_access: "Administrator"
   username: "jarodclark"
```

Figure 5: Local Storage

Course Roster Upload

One important specification for this project is being able to upload a course roster easily so there is a populated list of students that can receive a survey. The first problem to address is how the course roster is formatted to read in the data in an efficient and accurate way. Before implementing this functionality, it was decided that a student's information should include the following information; username, password, email address, first name, last name, Campus-Wide ID (CWID), and their team assignment. From this information the course roster is formatted to include all of this information for each student, with the addition of the class name and section the student belongs to. All of this information is held in a sheet with a .csv or .xlsx extension. Figure 6 is an example of course roster format.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Term Desc	Department Desc	Course ID	Section	CRN	Insturctor Full Name LFMI	Student Last Name	Student First Name	CWID	Email Address	Project ID	Project Name
2	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Abbott	Hannah	70723787	habbott@mymai	F21-15	Human Centered Design Studio (HCDS)
3	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Bagman	Ludo	70720128	lbagman@myma	F21-55	Next-Generation Renewable Energy Storage
4	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Bagshot	Bathilda	70639347	bbagshot@myma	F21-29.2	Amusement Park Thrill Ride Design - Team #2
5	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Bell	Katie	70712581	kbell@mymail.m	F21-16	Engineering for Communities Design Studio (ECDS)
6	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Binns	Cuthbert	70743375	cbinns@mymail.	F21-25	Commercial Wind Farm Design
7	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Black	Phineas	70721155	pblack@mymail.	F21-30	Hine Lake Fishing Pier
8	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Black	Sirius	70719489	sblack@mymail.	F21-22	Mine Waste Stabilization Design
9	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Bones	Amelia	70716462	abones@mymail	F21-42	Protective Coverings for Astronaut Gloves/Boots to Mitigate Lunar Dust Effects
10	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Bones	Susan	70730284	sbones@mymail	F21-25	Commercial Wind Farm Design
11	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Boot	Terry	70700887	tboot@mymail.n	F21-08	AIAA Design, Build, Fly
12	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Brown	Lavender	70685171	lbrown@mymail	F21-58	Downhole Tractor Optimization
13	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Bulstrode	Millicent	70721069	mbulstrod@myn	F21-29.1	Amusement Park Thrill Ride Design - Team #1
14	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Burbage	Charity	70617880	cburbage@myma	F21-15	Human Centered Design Studio (HCDS)
15	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Bryce	Frank	70693292	fbryce@mymail.	F21-22	Mine Waste Stabilization Design
16	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Carrow	Alecto	70641429	acarrow@mymai	F21-17	Smart Environments Design Studio (SEDS)
17	Fall 2021	Engineering Desig	EDNS491	A	81116	Csavina, Kristine R.	Carrow	Amycus	70718147	acarrow@mymai	F21-53	Digital Canopy for Monitoring Methane Emissions

Figure 6: Course Roster CSV

Now that the data is organized all in one place, the next problem to address is how to read in the data into the database to abide by the connections between classes, teams, and students. To accomplish this task, the class is added to the database through an application program interface (API) 'post' request first. After the request completes, the file is mapped to find team names. Each line contains a team name, and multiple students are assigned to the same team, so when a team name that has not been posted already appears, the team is added to the database with a foriegn key to the current class. Now that the current class has a list of teams, the file is mapped again to assign each student to the correct team checking if the project id assigned to each team and the project id located in the file match.

There are multiple edge cases to consider since the course roster can be a fairly extensive file in some classes like Chemistry 101 or Calculus 100. With a large file being read in, the API fetch requests take longer to complete, not allowing for the data to be properly stored before moving on to the next step in student creation. To solve this problem a timer is set to scale to the size of the file loaded in. As a file grows in size, the time allowed for fetch requests to complete grows as well.

Quality Assurance

Quality Software Plan

- On every update to the code, check to make sure front-end and back-end compile
  - Continually run the program after making small changes to ensure that the code is still compiling
- Python unit tests for database structure and functionality
  - Questions can fill topics through foreign key
  - Students fill teams, teams fill classes, classes fill departments
  - Category descriptions fill categories, categories fill questions, questions fill topics, topics fill surveys, surveys fill classes
  - Categories and questions have list fields inside the database, so making sure these lists can be properly implemented to the database
- Weekly client meetings to check whether format and functionality are acceptable
  - Each week updates have been done and the client has given feedback
  - Features and colors have been checked and there has been outside feedback from family/friends on what looks pleasing to the eye or not
- Abundant use of console logs to debug interface
  - When a bug is found, console logging to find and get rid of the bug
- Use CSV files for load and stress testing
  - Large files were given by the client to run stress testes to the program
  - CSV files are loaded to and from which will be checked to ensure the right number are coming out
- Code metrics are based on user interface and making sure back-end functionality is connected
  - Ensuring that when an implementation to either end, the other end is updated in real time
  - Posting, updating, and deleting from front-end interface will also update the back-end database in real time
- Static program analysis to determine formatting and design techniques
  - Using online documentation to determine what looks good
  - Using outside perspectives to help with feedback
  - Ensuring user accesses are handled properly
  - Refactoring along with finding any vulnerabilities in the code
- Checking edge cases
  - User entered in more than once for different departments
  - CSV files in specific formats
- Test complexity and speed through back-end python tests and UI testing through website compilation

Results

Performance Testing & Summary Testing

The front-end and back-end of the project are tested as certain functionality parts are implemented. The back-end is tested through python-unit-tests. These python tests run through the database and ensure that any connections through a foreign key or a many-to-many field are correct and working appropriately. These tests ensure that the database is made correctly and every feature has the attributes that it is supposed to have. Next, the front-end is tested through compilation tests and inspections on the web page. The tests are on Google Chrome since that is the most popular web browser but the tests are also tested through Safari, Microsoft Edge, and FireFox since these are the next most popular browsers. This is done by inspecting the web interface on every page click through and action to make sure no errors are occurring below the surface.

Results of Usability Tests

A few usability tests were conducted through the software development process. One usability test that was performed was weekly live demos with the client. Since the client has used the previous software and will also be using the new software, ensuring that the software was user friendly and had the functionality needed was important. These live demos allowed the software to get feedback on functionality that had recently been added so going through and making changes wasn't as difficult. Another usability test was allowing non-computer science students to play around with the software ensuring the software is user friendly to a different audience along with making sure no new bugs are found.

Future Work

Most of the project functionalities have been completed. The two main functionalities that have not yet been implemented are the results in front-end and the back-end database for the survey and the automated emails to notify students when the surveys become available and superusers when the results have been sent to them. All the other future work is little implementations and small requirements that the client wants.

Future Work List

- Implement the grading system based on the number of questions, teammates, and which button is selected in the rows
- Allowing the survey results to be sent back to the superuser
- Allow the superusers to send certain comments back to the students
- The results are put into an easy to read CSV file based on the multiple choice
- Design and implement automated emails
- Allow for the software to run on all types of computers
- Adjust the web application to be able to fit on all types of windows and devices
- Ensure that professors can be an admin in one department and a superuser in another
- Implement Mines Multi-Pass to replace current login page
- Finalize and get the web application servers running

Appendices

- CATME
    - Comprehensive Assessment of Team Member Effectiveness
- IDE
    - Integrated Development Environment
- EDNS
    - Engineering Design and Society
- CWID
    - Campus-Wide ID
- CSV
    - Comma-separated values
- UI
    - User-interface
- User
    - Student
- Superuser
    - Professor
- Admin
    - Course Coordinator
- Django
    - Web framework that allows for a clean designed database
- React
    - Javascript library that allows for an interactive UI
- API
    - Application Programming Interface
- JSON
    - JavaScript Object Notation