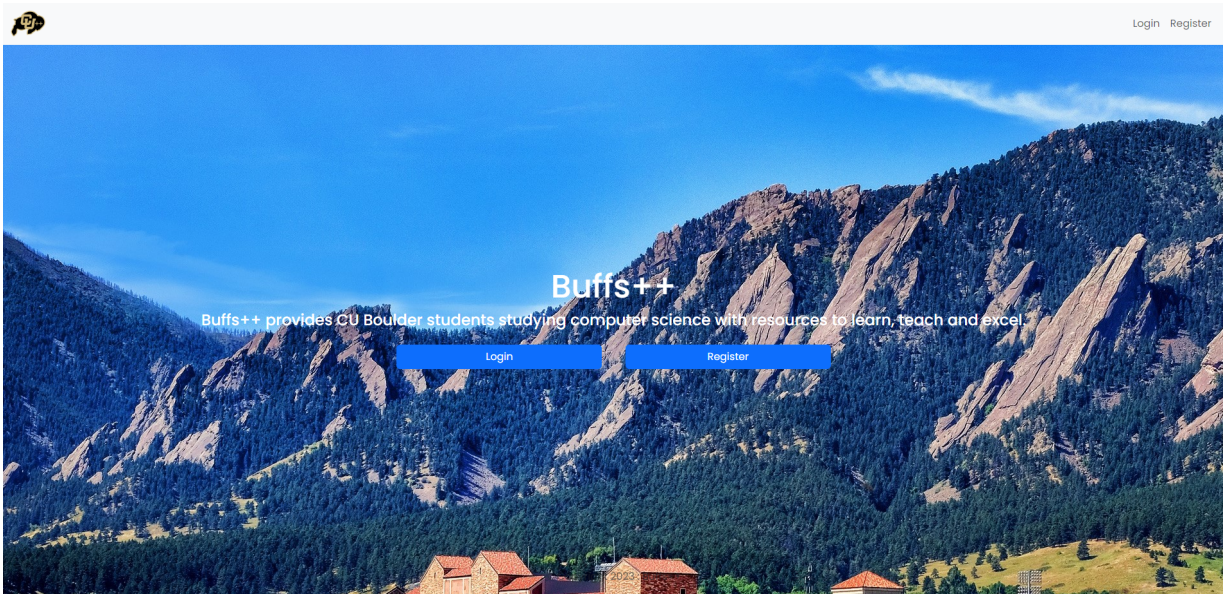
Buffs++ Tutoring

The Buffs++ Team: The contributors of the Buffs++ Tutoring project were Amelia Lunn, Cade Mather, Sophia Soka, Nick Cisne, Ojas Baral, and Vesaun Shrestha, and we worked hard to create a fully functioning website connecting CU Boulder Computer Science students to tutors.

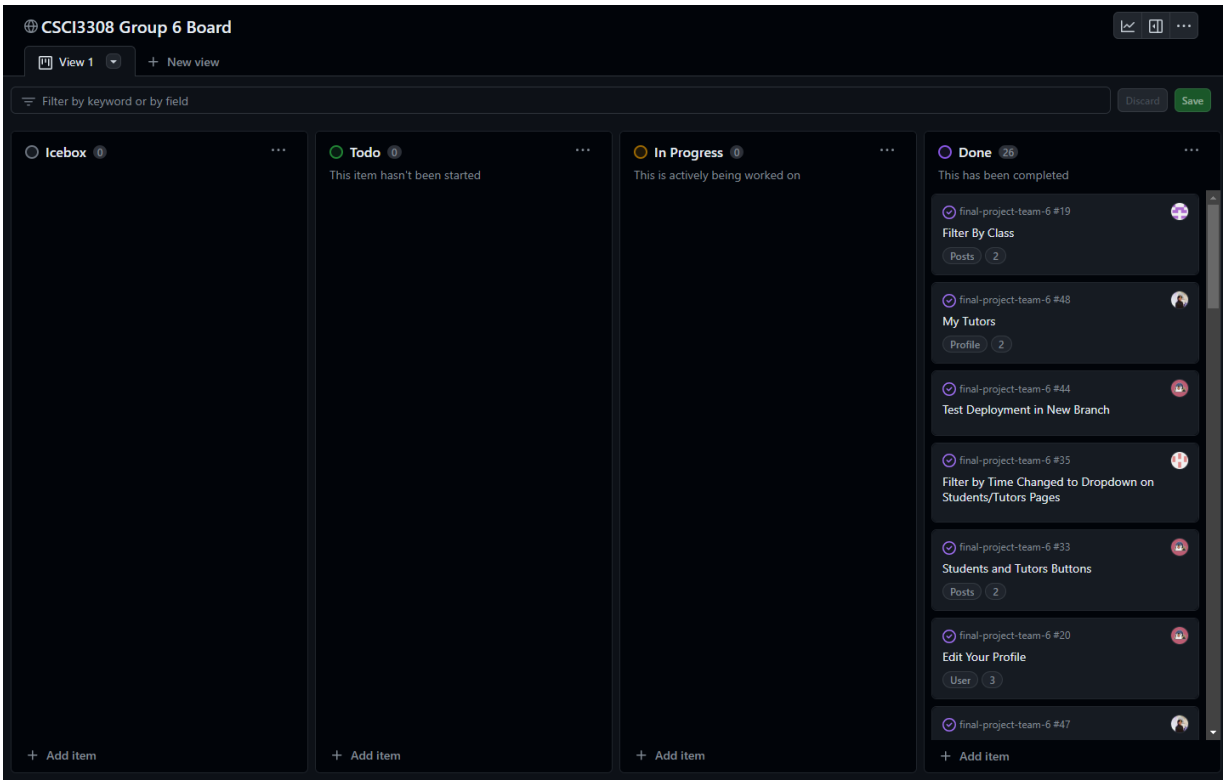
Project Description:

Buffs++ is an exclusive tutoring application for computer science students at the University of Colorado Boulder. Buffs++ provides these students with a platform to find tutors, or for tutors to find students while building the computer science community at the University of Colorado Boulder. Through secure registration and login functionality, users can build their profile with information about themselves, whether they are a tutor and/or student, the classes they are tutoring/looking for help with, and other information to help match users. Our application has two user friendly feeds, the Student feed, and the Tutor feed. On the Student feed, users can see all users who are in need of tutoring for a specific class. On the Tutor feed, users can see all users who are available for tutoring in a specific class. Both these feeds allow users to filter results by name, email, classes, and time available. Buffs++ is built for the CU Boulder computer science community, and therefore allows students to network by creating relationships. A tutor user can create a relationship with a student user, or a student user can create a relationship with a tutor user. These relationships can help users keep track of who they have worked with, and boosts credibility for students and tutors. Our application has the potential to enhance and grow the CU Boulder computer science community, so join us and register today!



Our Website's Landing Page

Project Tracker:
Our Project Board: https://github.com/users/ojasbaral/projects/1



Video: https://youtu.be/P_oueoYjEvw

VCS:
Our GitHub Repository: https://github.com/ojasbaral/final-project-team-6

Individual Contributions:

Ojas Baral: I worked on the following features:

- Landing Page
    - Using HTML/CSS and Bootstrap, I created a landing page that users will be directed to when first going to our site.
- Profile Page
    - I rendered a user's information on the profile page, and also gave users the option to start or stop tutoring a class.
- Create Relationship
    - Users are allowed to create a relationship with another user, given that the relationship is valid.

- My Students/Tutors
    - I gave users the option to view their own tutors and students based on their relationships.
- Login, Register, and Update Profile Tests
    - I created both positive and negative unit test cases for the Login, Register, and Update Profile tests.

Cade Mather: I worked on these features:

- Populate Profile Page
    - Used Html, Bootstrap, and Javascript to create text boxes and dropdown menus for the user to input information into their profile
    - I made it so when the user inputs information to their profile it updates the SQL database
- Students/Tutors Page
    - I implemented dropdown menus on the search bar on the students and tutors pages that allows the user to select courses and/or times that they want to see either students/tutors for

Nicholas Cisne: I worked on the following features:

- Login Page
    - Using EJS, I created the login page that checks if the user is in our SQL database, altered from Lab 9 to expect an email string instead of a username.
- Register Page
    - I created the registration page using EJS, allowing users to sign up with their email address and a password of a specified format. Both fields are required, error messages pop up if an email has previously been registered to our site. Passwords are hashed using the Bcrypt library via NodeJS.

- Deployment
    - Created the virtual machine using Microsoft Azure and installed all of the dependencies (Docker) to host our site.
- Students/Tutors pages
    - Helped work on the functionality for both of these pages, including updating/querying our database and properly rendering certain components.

Sophia Soka: I worked on the following features:

- Students page
    - Created both the front-end and back-end of the students page to display all users in our database that were students, displaying their email, class and times as well as a button linking them to their profile page.

    - Connected the database to our search endpoints in order to filter the list displayed
- Tutors page
    - Created both the front-end and back-end of the tutors page to display all tutors in our database, displaying their email, class and times as well as a button linking them to their profile page.
    - Connected the database to our search endpoints in order to filter the list of tutors displayed
- Search Bar
    - Created the first version of our search bar (with all fields (name, class and times) all being text boxes) on both the students and tutors page using Bootstrap, HTML, SQL and Javascript.
- User Acceptance Criteria
    - Wrote out all positive and negative test cases for each endpoint in the User Acceptance Criteria doc and made sure each feature passed the acceptance criteria manually

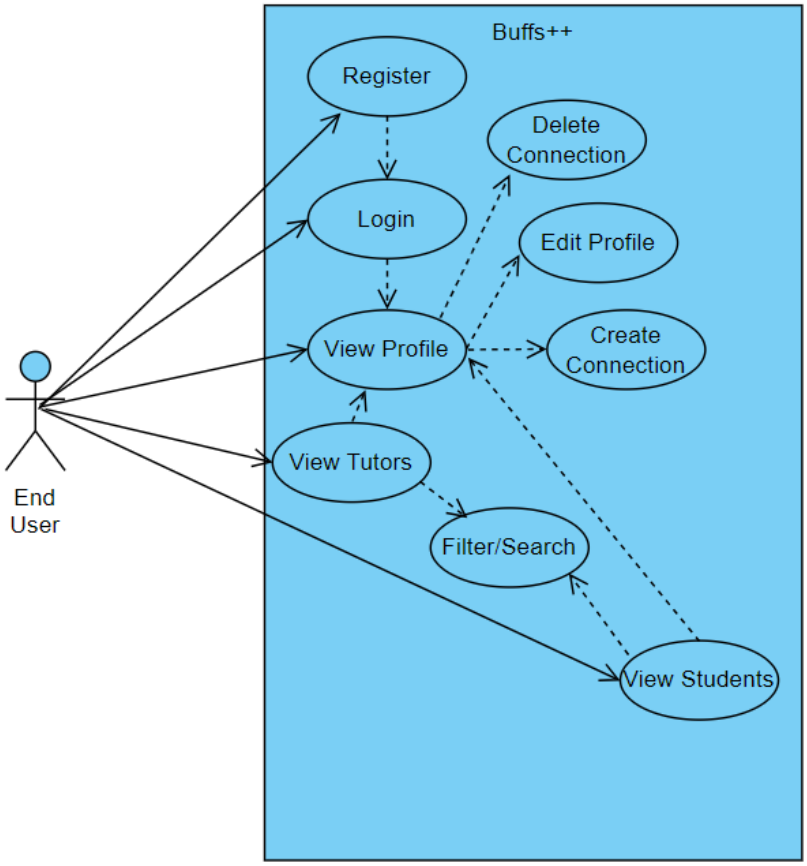
Vesaun Shrestha: I worked on the following features:

- Endpoint Test Cases
    - Used Chai/Mocha framework in order to test the functionality of our APIs.
- Students/Tutor Pages
    - Helped design and implement the EJS files for our student and tutor pages.
- Bug Fixes
    - Late stages of our development presented minor miscellaneous bug fixes that included removing and adding certain features and adjusting the styling of certain text on the students/tutoring pages.

Amelia Lunn: I worked on the following features:

- Release Notes
    - Created and maintained an ongoing record of the project releases that aims to track all updates, changes and new features and bug fixes throughout the process.
- Students Page
    - I helped with the functionality for the students page on the backend and the frontend, working in and out of the database to create filters for the classes.

Use Case Diagram:



Test Results:

Feature	Information needed	Test cases	How to test	Results
Login	- Email - Password	Email not found in users: Redirects to register page Wrong Password: Throws an error message and reloads login page Correct email and password: Redirects to home page	PostgreSQL database has been prepopulated with dummy data concerning users. Therefore, for the positive test case, we will use the user created during the /register test case. For the negative test case, we will use a user from the dummy data with an incorrect password.	All mocha and chai test cases pass.
Register	- Email - Password	Invalid email: Message asking user to input valid email Email is already in the users query: Throws an error message and redirects to the landing page	PostgreSQL database has been prepopulated with dummy data concerning users. Therefore, for the positive test case, we will use new data not already in the database . For the negative test case, we will use invalid email addresses, and emails and	All mocha and chai test cases pass.

		Unique email and password: Insert a new user with the information given into the user query and redirects to login	usernames that already exist in the database.	
Populate Profile	(Not needed but accepted) - Bio - Classes - Times Available - Contact Info	Nothing is entered: Nothing is changed and there's no errors If user inputs invalid time or classes: Error message appears for user User inputs all valid information: Information in users table is updated and site returns to profile page	PostgreSQL database has been prepopulated with dummy data concerning users. Therefore, for the positive test case, we will use new data not already in the database . For the negative test case, we will use invalid classes and times available that already exist in the database. We also have created a test case for the positive and negative situations in this scenario.	All mocha and chai test cases pass.
Create Relationship	- Which user is the tutor - Which user is the student	Session user is the tutor: Row created with end user as the tutor and other user as student Session user is the student: Row created with the end user as the student and other user as a tutor	We will test this using user acceptance testing, we created the Acceptance criteria in the database and will follow it.	User easily navigates creating a relationship and the page does not allow the user to create a false relationship. User doesn't deviate from expected actions. Test case passes.
Delete Relationship	- Which user to delete a relationship with	End user selects a user to delete a relationship with: Row is removed from the table	This is tested via the Acceptance Criteria and therefore will be tested manually.	User easily deletes a relationship and the user doesn't deviate from expected actions. Test case passes
Search	(Not needed but accepted) - name/email - Class - Times Available	No information added to any box: Returns all profiles that are tutors (or students based on which page) One search box filled with info: Returns all profiles that have similar info compared to the box it was entered in Multiple search boxes used: Compares multiple fields with respective search information. If no profiles match, page is blank	This is tested using the Acceptance Criteria and will have to be done manually.	User easily navigates searching and the page reacts appropriately based on user inputs. User doesn't deviate from expected actions. Test case passes.
Tutors	-user_id is logged in	User is on tutors page without entering anything in search bar: Displays all current tutors in the database with links to each tutors profile page User is on tutor page with info in search bar: Refer to Search feature above	This will be tested manually using the Acceptance Criteria and the premade database. Positive test: If all tutors are displayed with working links to their profile because user is logged in Negative test: Page isn't available to view because user isn't logged in yet	It is easy for users to find and navigate the tutors page. Users don't have access if they have not yet logged in. Both positive and negative test cases pass.
Students	-user_id is logged in	User is on student page without entering anything in search bar: Displays all current students in the database with links to each tutors profile page	This will be tested manually using the Acceptance Criteria and the premade database.	It is easy for users to find and navigate the students page. Users don't have access if they have not yet logged in.

		User is on students page with info in search bar: Refer to Search feature above	Positive test: If all students are displayed with working links to their profile because user is logged in Negative test: Page isn't available to view because user isn't logged in yet	Both positive and negative test cases pass.
View profile	-user id of profile you are viewing	Current user is connected with other users profile: User is rerouted to other users profile with delete relationship button in top right corner Current user is not connected with other users profile: User is rerouted to other users profile with create relationship button in top right corner	This is tested by comparing it to User Acceptance Criteria and will be done manually.	Users change their profile in whichever way they would like and profile updates accordingly. Users don't have access if they have not yet logged in, and users can't deviate from normal and expected actions. Test case passes.

Deployment:

Our Website Link: http://recitation-12-team-06.eastus.cloudapp.azure.com:3000/

Our website is hosted on a virtual machine, using Microsoft Azure as the hosting service. It is currently running on a Docker container on this VM. To access the site, simply click on the link above, where you will be redirected to our site's landing page. From there, you can register an account with us, login and explore what our application has to offer.