John Gay

Jonny Koppelman

Wilson Wong

***Tutor and Student Connection***

***(TaSC)***

**Overview**

Many students want and/or need tutoring, especially at a rigorous school like RPI. However, the tutoring system here, ALAC, is insufficient, as many students are unable to get the tutoring help they need due to time constraints or unfortunate circumstances (such as too many students on the queue). This leaves those who need help without it. Students can go to their professors to ask for help or request a tutor, but this also takes up time. We designed this application with these problems in mind to create a better system to facilitate tutoring across RPI. This will get rid of the unneeded steps of involving an intermediary such as a professor or the ALAC coordinator by creating direct connections from student to tutor.

**Aim**

The purpose of this application is to connect students to tutors and vice versa in order to help make tutoring more widespread. With direct connections, students and tutors can establish a time for one-on-one tutoring for students who need more attention than those who go to office hours, which are typically crowded. A forum also allows students to post questions which do not require a full office/tutor visit and can be answered relatively quickly by both students and tutors.

Some challenges such an application faces are popularity and simplicity. This application’s success is determined by social impact, as its usability will be entirely based upon the usage by tutors and students. Because of this, the product must be fully functional before launch. This also ties into simplicity. Users desire an application that is easy to use and brings many benefits. This requires a delicate balance between functionality and simplicity.

**Application Functionality**

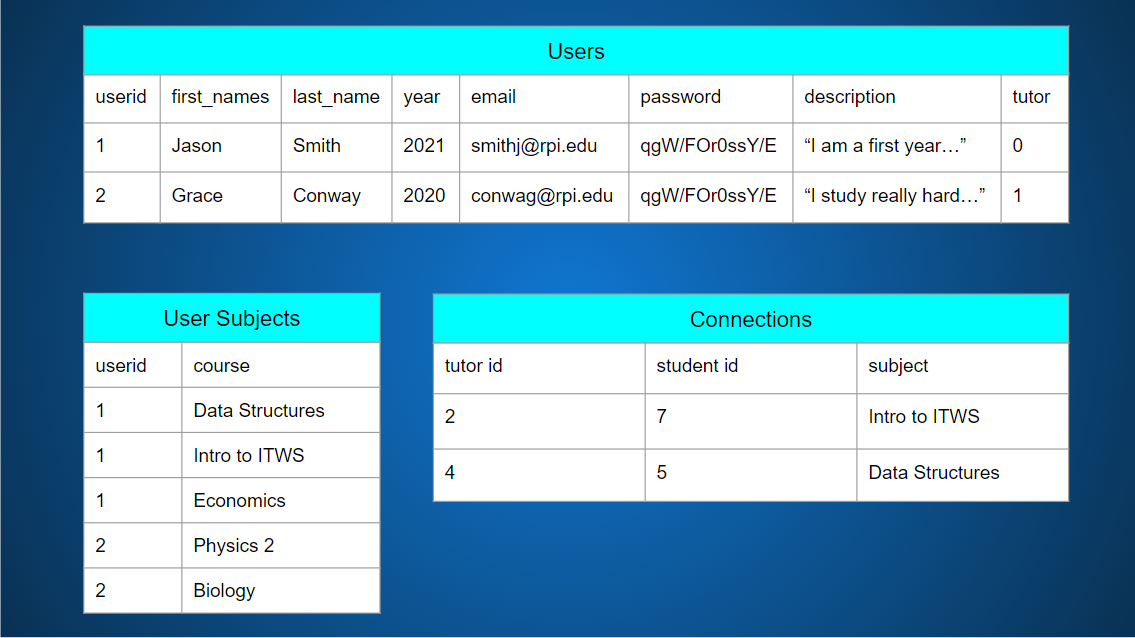
In this application, we focus on putting and pulling real information from and into a database. All aspects of our project relies on the database in order to create dynamic php web pages that react to user input in real time. To create a new account, a user can click the link to sign up on the login page, which brings them to a new page to submit information. The user has to enter information about themself and set a password. The user can then choose whether to be a student or a tutor, and when they are done they are brought back to the login page. To log-in, the user enters their email and password, and then the user input is queried into the users table. If no user is found with that email and password, a message is displayed stating wrong email/password. If a user is found with the matching email and password they are brought into the connect page with a session storing the user-id.

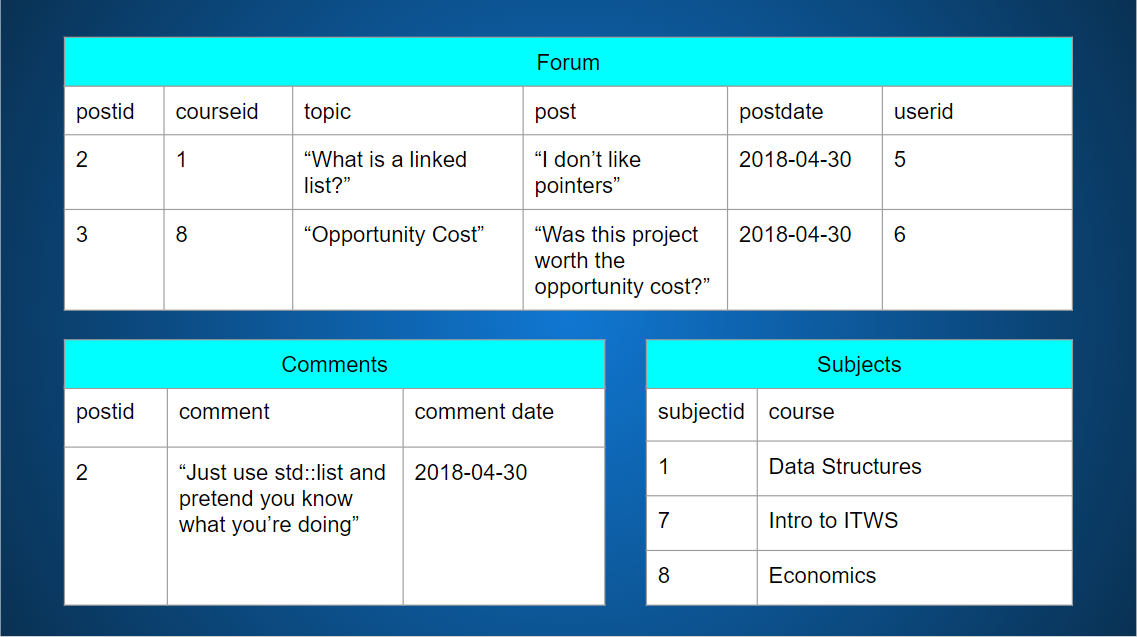
The connection page uses the connections table to load all of the user’s connections. The user-id is found through the session created upon login, and is accessible in all pages. The connect page queries into the connections database and finds the ids of all users in the column opposite of the user. In other words, if the user is a tutor it finds all student ids where the tutor id matches the user id, and vice versa otherwise. Then the information of each connection is pulled from the users table and user\_subjects table to print out the information from each connection. If a user wants to make a new connection, they can click the “Make a Connection” link on the sidebar. They then can enter a subject to search for all users in the subject that have the opposite boolean value in the tutor column, meaning they will only find tutors if they are a student or students if they are a tutor. Once the user searches for a subject, they are given a list of people for the subject, where they can choose to connect with each person. When they press the connect button, the tutor id and student id are added to the connections table through the connectionmade page and a success message is displayed on screen. On the sidebar they can then go back to the connections page, and will see the newly made connection.

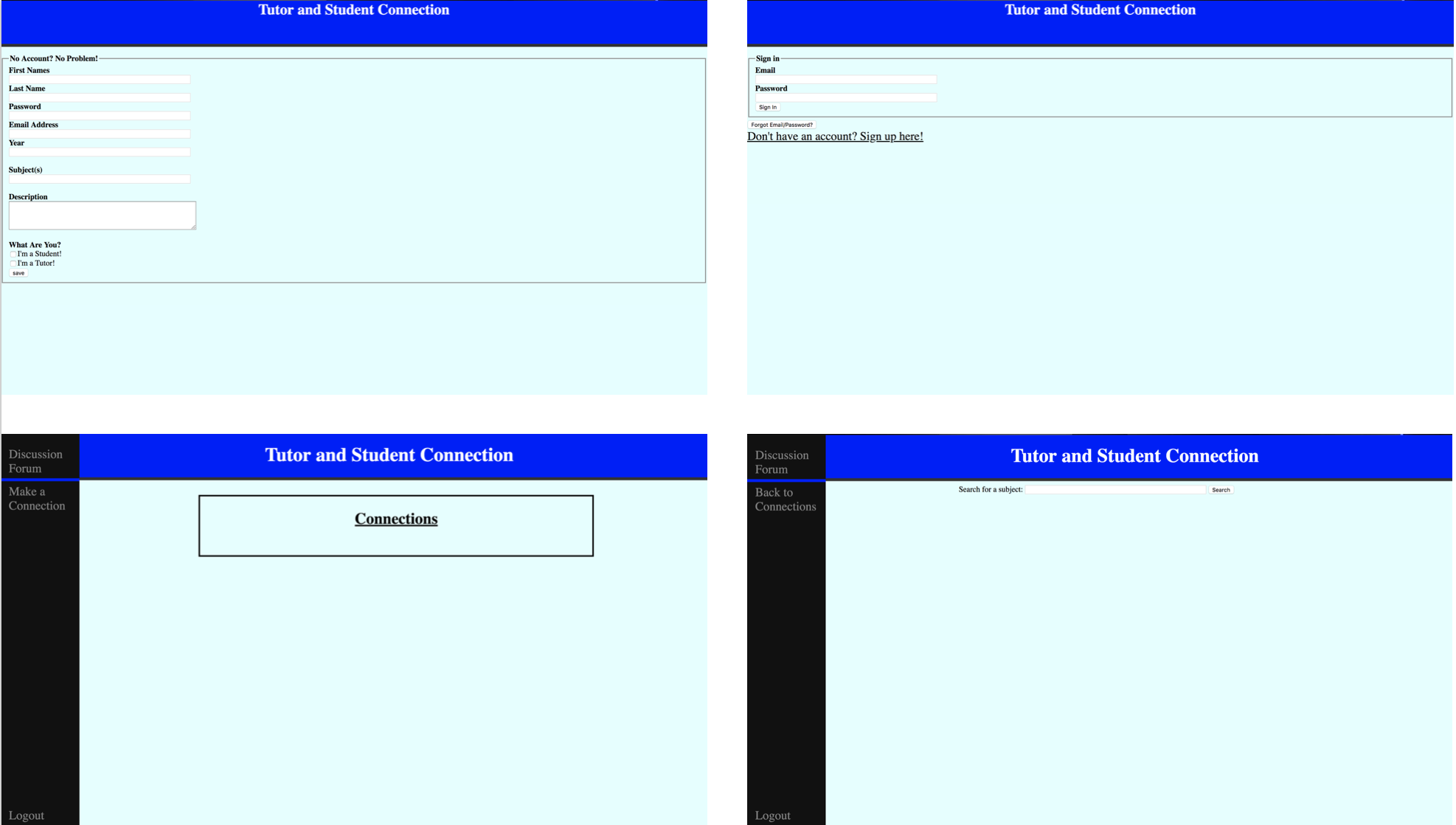
The forum page dynamically loads all of the user’s subjects into a sidebar by checking the database for all instances of the user’s id in the user\_subject table. This then allows the user to partake in posting, viewing and commenting on threads in all the subjects the user is signed up for. The database stores all of the threads in one table called “forum,” and displays them in their respective subject dynamically, by a simple check to see if the current selected subject id matches the subjectid stored in the forum table. Once a thread is selected, all comments on said thread are displayed by doing a similar check. The check is to display all comments, stored in the “comments” table, that have the same postid that the current thread contains in the “forum” table. To make a comment, once save is clicked, the php file posts the comment in the “comments” table with the current comment post id, and the comment itself. Then the page reloads and the new comment appears in real time.

**Table Schema**

In order to store the all of our data in relational tables, we created 6 tables to keep track of all the users’ data. The users table holds the information of each individual user. It has a unique user id for every user, along with first names, last name, year, email, a short description describing the user, an encrypted password, and a boolean value representing whether the user is a student or a tutor. This information is stored and printed out whenever the site needs to describe a student, which can show to which connections to make and what the connections a user has already made are. The user\_subjects table is used in relation to the users table, relating each user to the subjects that they have signed up for. The user id can be repeated throughout the table for users who have signed up for more than one subject. The connections table keeps track of each connection made on the site. It contains a tutor id and student id to relate the two users and their roles, along with a subject that corresponds to what course is being tutored. The forum table is used to keep track of each thread that is created. It contains a post id, a unique identifier for each post, a course id, to correspond each post to a subject, a topic for the post, a post description, a date for when the post was created, and a user id for the user that made the post. The subjects table is a relational table to match each course with its id that is held in the forum table. Finally, the comments table keeps track of all the comments on the posts. It contains a post id relating to the unique post id in the forum table, a comment where the user can say what they want, and a comment date. Because there can be multiple comments on a post, the post id can be repeated throughout the comments table.





**User Functionality**

To use this application, a user logs into or creates their account, allowing them access within the application. The user can then search for a subject they are looking to tutor/find help in, and a list of tutors/students in that subject will be displayed. To connect with the other person, the user can click on the connect button, which will bring them to a page displaying connection made. Once the user goes back to their connections, they will see their new connection displayed.

Alternatively, a user can visit the forum to comment on, search for, or create a thread. The threads are organized by subject, which can be selected on the sidebar, so that each subject only displays its respective threads. The threads themselves are then further organized by date, with the newest on the top. Once a thread is selected the user can participate in the conversation by posting their own comments. Students as well as tutors have access to this functionality.

**Challenges**

Throughout this project, there were many difficulties that our group had to overcome in order to deliver our successful product at the end. The first large challenge was the fact that our project’s functionality depended entirely on php and sql, which we were unfamiliar with. We knew that we would learn it, but we would have little time left to implement it if we waited until then. To compensate for this, we made our html and css early on. Instead of creating mockups through drawings or other tools to design a website, we created all of the html and css files that we needed as our mockups. This way, we already had the design and layout of our website completed, so we would only have to worry about the back end. Additionally, we tried dynamically building our website using ajax and a json file. This gave us a template for what we needed to do once we began the php. Also, our original secondary focus was on javascript, but we quickly realized that our main focuses were pushing and pulling information from and into a database. Finally, once we felt comfortable using php, we began to implement the back end of our site, and managed to successfully create all functionality we intended to.

Another challenge we faced was how to create a login capability. At the time, we did not know of any external libraries we could use to handle this for us, so we implemented our own tables to hold user information. To make the login work, we stored the user id in the session once logged in, so the webpage would dynamically build the site based on the connections and subjects the user was in.

**Future Goals**

We have a wide array of features that we would like to implement in the future. The first is javascript to increase intractability, such as drop down menus instead of search bars so we do not have to depend on user input. Another feature is having a functioning mobile platform, because this app could come in handy for users to quickly access on their phone. We would like to implement a rating system for tutors, so students can read peer reviews and discover which tutor suits them best. We plan to create a voting system for posts as well, where posts and comments can be upvoted and be placed higher on the lists, and to also favor tutors who answer more questions and get more upvotes.

The main upgrade we would like to make is to make the app more personalizable, so that each student and tutor can make their profile how they want. We want to add a feature where users can add or remove subjects, because as students progress through school they will need tutoring in one course in a semester and a completely different course the next semester. Similarly, tutors will gain more experience and be able to tutor more courses as they progress through school. To keep tutors from getting too many students, we want to keep track of connections made and time spent tutoring for each tutor, so that no single tutor has too many connections being made where they become overwhelmed. Lastly, we would like to make each user sign up as both a student and a tutor if they please, because while someone may need help in one course, that does not mean they are not able to tutor another course.

**Conclusion/Reflection**

After completing this project, we are extremely satisfied with our end result. We successfully built our web application to dynamically build a personalized page for each user that logs in. This means the app loads the connections that that user has made, will only display other users of the opposite boolean tutor value in the search page, and only make the threads for the subjects that the user is enrolled in. We learned a lot about how websites work and how the back end is developed, because we used php for the entire project and gained a great deal of experience in it. This project taught us a lot web development, helped us learn how to work as a team to create different aspects of our application, and gave us a great project that we can use and show and even continue to work on in the future. Overall, we are pleased with our performance in this project and thankful for the opportunity to build it in this class.