The theme of my project 3 is Your Favorite Cornell Course. I have always been interested in exploring the great variety of courses available at Cornell, but because of the limited time and energy I have, there are only so many courses I can experience myself. Therefore, I want to use this opportunity to conduct a survey on what classes my fellow Cornell students love. The target audience of the project is naturally Cornell undergraduates who have taken courses here. The major goals of the website is to provide a platform for Cornell students to both share and hunt good or interesting classes.

In the form, I put in 9 items – name, email, year, college, class title, professor, easiness, workload, and comment. The first four are all about personal information. I included year because a senior's suggestion/recommendation would intrinsically be more credible and serve as a better reference than a recommendation from a freshman, but this is not an essential item since most people do have very different personal preferences when it comes to fields of interest, so I put it down as optional. Whereas the item college within Cornell is required because Cornell is so big that students from different colleges have little in common in class schedules and academic paths, which makes the favorite class of a student from, let's say CALS, remotely relevant to a student from Engineering. Therefore, after a post is submitted, aside from highlighting all the posts of the same user, my program searches for posts from the same college and display these posts separately to give the current user a more tailored result. Five out of the nine items are asking for information about the class. First of all, the class number and name are the foremost information to collect. This is the most accurate way for the viewers to know which class the post is about. In order to further reinforce the formality of provided class number and name, I provided an example for the posting format, and in the php program, the program would only pass a variable that is at least ten characters long and contain both letters and numbers with letters preceding numbers (because that is how normally class number works). Based on personal experience, sometimes it is not how the class is, but how it is taught by the professor. A seemingly boring class can turn out to be fascinating because of the professor's teaching style and personal charisma. Therefore, I included the professor item, though as optional since this is not a guaranteed factor in people's liking of classes. Easiness is another potential factor that comes into play quite often when it comes to whether someone likes the class or not. It is rare for students to love a class that is ridiculously hard and tedious even though it might turn out to be very useful in the future. However, like I said, this is not guaranteed, so I didn't not make it required. Workload, on the other hand, is a different story for Cornell students. I understand that everyone has much work on hand, so it is important to provide at least an estimate of the class's workload for viewers to consider. Because some classes can be interesting and heavy-loaded at the same time, and it would be unreasonable for someone to take this kind of classes, that require a big chunk of people's time and energy. Finally, the form asks for comments/reasons, which has to be longer than three characters. This is to give the user a less constrained way to express their thoughts on their favorite classes and further elaborate on why these classes are their favorites, which is the most important part of their posts. For students who are looking for good classes to take, the comment/reason item provides the most relevant information and valuable first-hand experience, or even suggestions on how to excel in a particular class.

Because this website is Cornell-based and solely targeting at Cornell students, I chose to use Cornell red and white as the color scheme to create a Cornell-like visual appeal. I also repeatedly used Cornell-related elements like logos, campus picture, and Big Red to reinforce the overall impression. All my labels and elements are vertically aligned to indicate clear association with each other. I indicated whether an item is require or not by adding notes right next to the item descriptions to assist users with completing the form. I utilized repetition in font choices to create visual unity as well. When highlighting current user's post(s), I used Big Red as the background image and black background color to attract more attention. The navigation buttons – Submit and Post and Posting Wall – are always at the top of the page, sticking out the frame to indicate significance.

For the course number item, I provided a complete example of how the form is expecting the course number to be like. For easiness and workload, I chose to ask for integers as estimation of both, but workload varies a lot more than easiness in general, so I decided to use input type range for workload, not input type number, so that users can provide more accurate estimations based their subjectivity. I also used textarea for the comment/reason item instead of text input to communicate the desired amount of content requested. All these form designs aim at reducing users' errors and obtaining expected information. As for error handling, I have multiple separate if statements to validate different items after the form is submitted. Instead of giving one general error statement for all error types, my program provides very specific feedback on what went wrong and how to fix them so to have an affective error handling system for the form. For example, there is an if statement specifically for checking whether there are special characters in username, professor, and course number. If a user submits a professor name like "David Gries…", the program will display a message that says "Your post is invalid. Please fill the form again and make sure that there are no special characters in your answers". The invalid forms will not be displayed, as a sign of failed submission. All the error notifications will be at the top of the posting wall to be visible and hard to miss. These notifications also have a different background color to remind users of the occurring error. With the Submit a Post button at the top, users can easily go back and fix the errors. If a form is submitted successfully, the program shows a success confirmation, like "thank you for submitting a post", as the feedback.

There are variations in what information is displayed for posts in under different sections. For the just submitted form of the current user, all fields are displayed except for email because email is more of a personal information that should be posted at serious discretion. For the current user's multiple posts from the past, only the course number and comment are displayed because it would be redundant to repost the personal information of the current user. For posts in the "same college" section and post history, only the required items are displayed because the space for displaying number of posts is limited and we should pick the most important ones to show. On the welcome page, I included a link to Cornell's course of study because sometimes people forget about the exact name of the class or the course number of the class and the link would make things more convenient.

Lastly, here is how my code is constructed.

1. Include the database and the common header file.
2. Check if a form is submitted or not by checking if all fields are empty. If any of the fields is not empty, the program takes it as a form was submitted. Validate each field according to respective requirements, and display error messages accordingly if requirements are not fully met. Mark the post with problems as invalid for later reference of the code. Only stores the post if it is valid. If there's no submission, which means the user is accessing the index page directly, display the welcome message. I wrote an extra function to check if the course number is consisted of letters and numbers with letters preceding numbers.
3. Display current post. Search the array of posts except for the latest post for posts with the same username, and display those posts.
4. Search for posts from the same college, and display those posts under current user's posts. If there's none, provide links of facebook and twitter and encourage them to invite their friends.
5. Display every post stored in the database.
6. Include the common footer file.