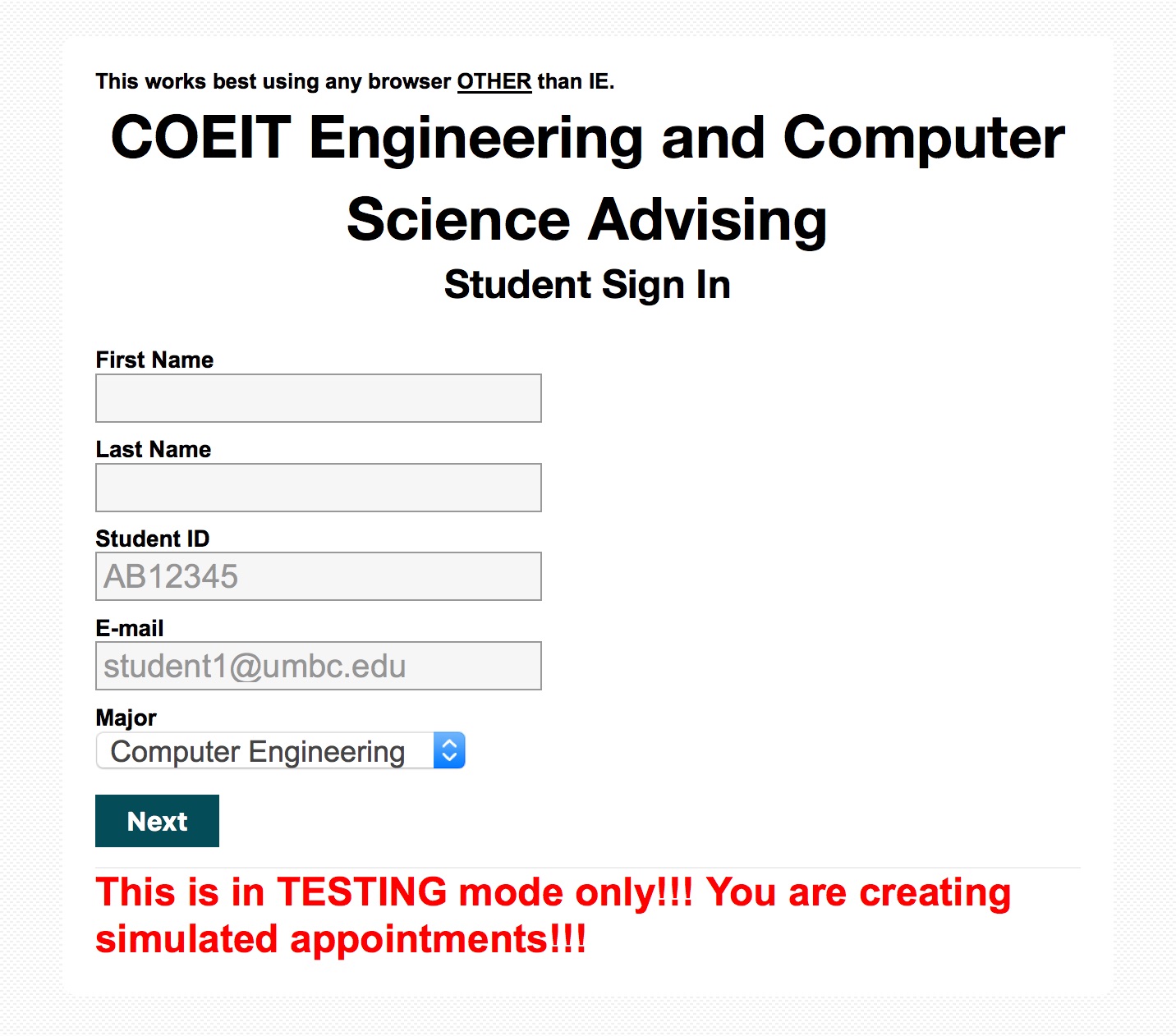
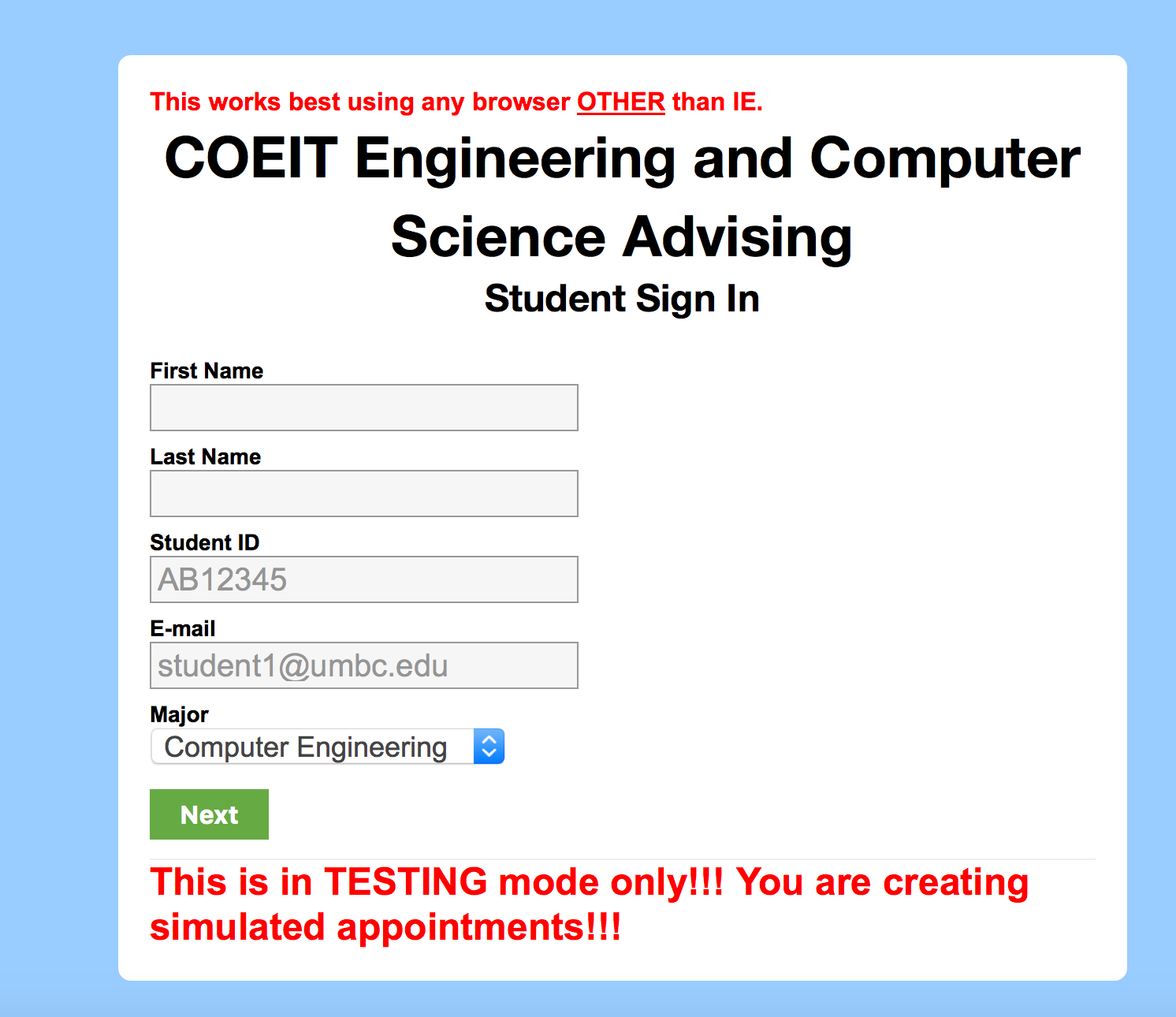
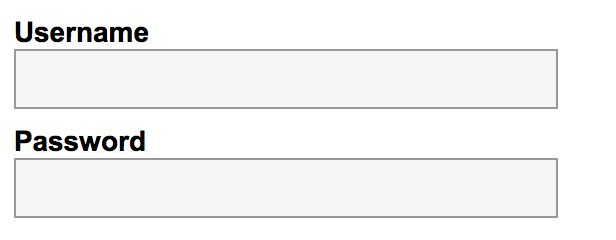
*UMBC CSEE Undergraduate Advising Project- Part 2*

James Bilbrey Luke Carrico Benjamin Nace

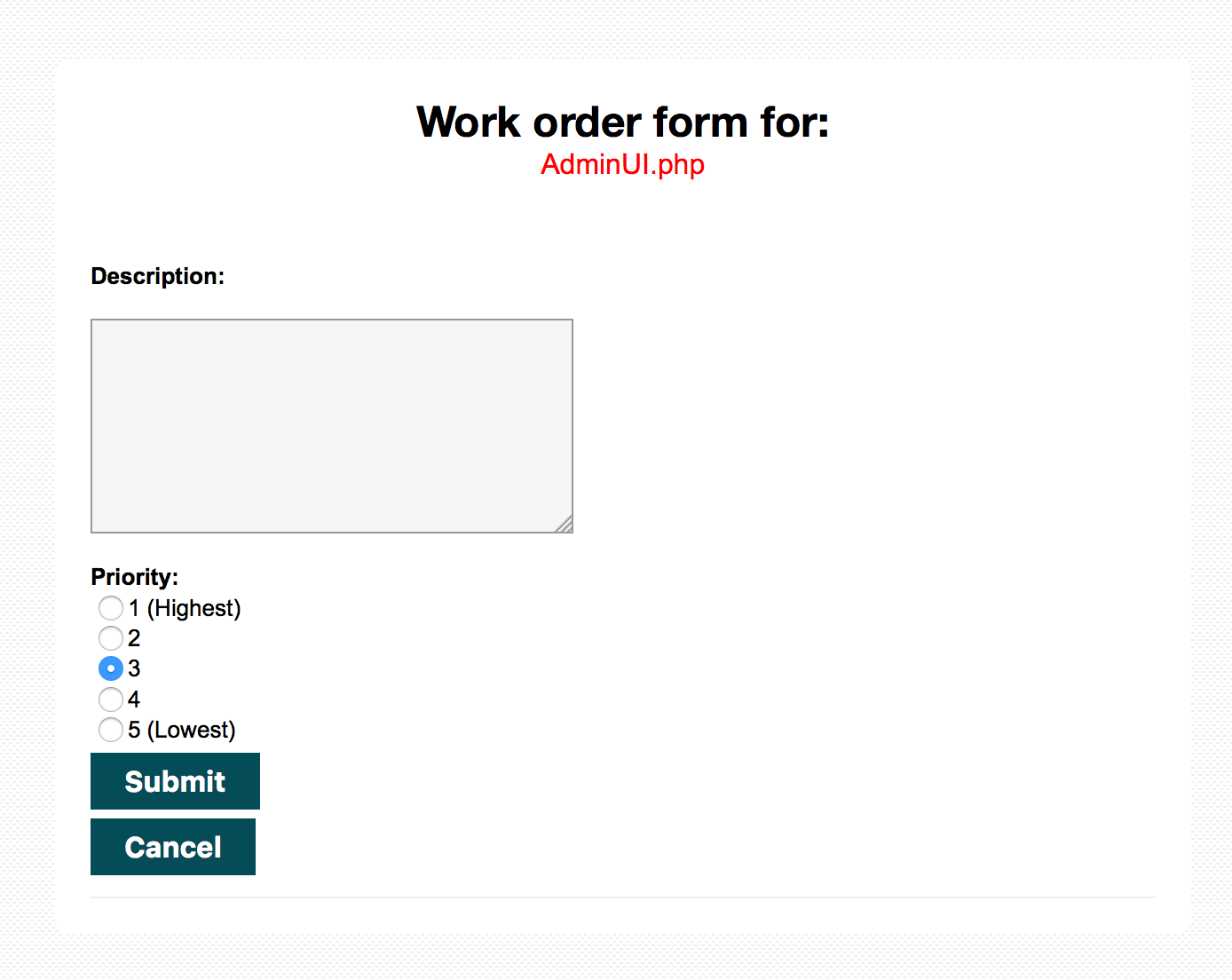
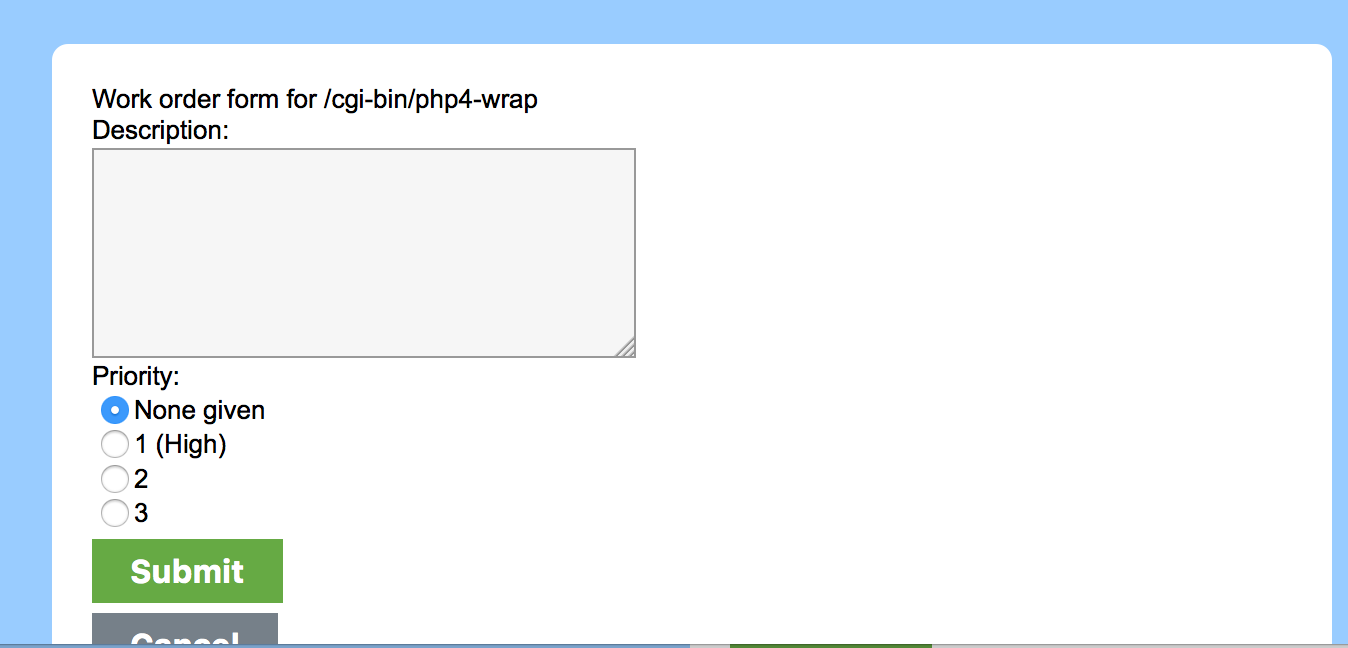
**Part 1: Advising Webpage**



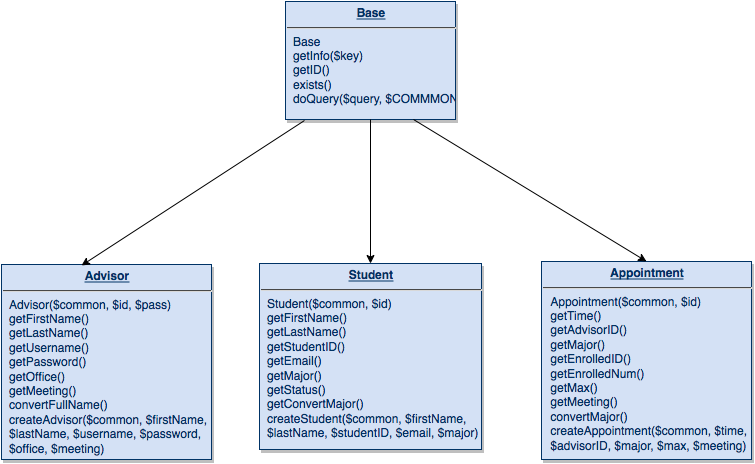
One of the changes we made was we changed the CSS to look more like the UMBC COEIT page. We used the same background image as the COEIT page, set the navigation bar color as our button color, and used the menu bar color as the button highlight color.



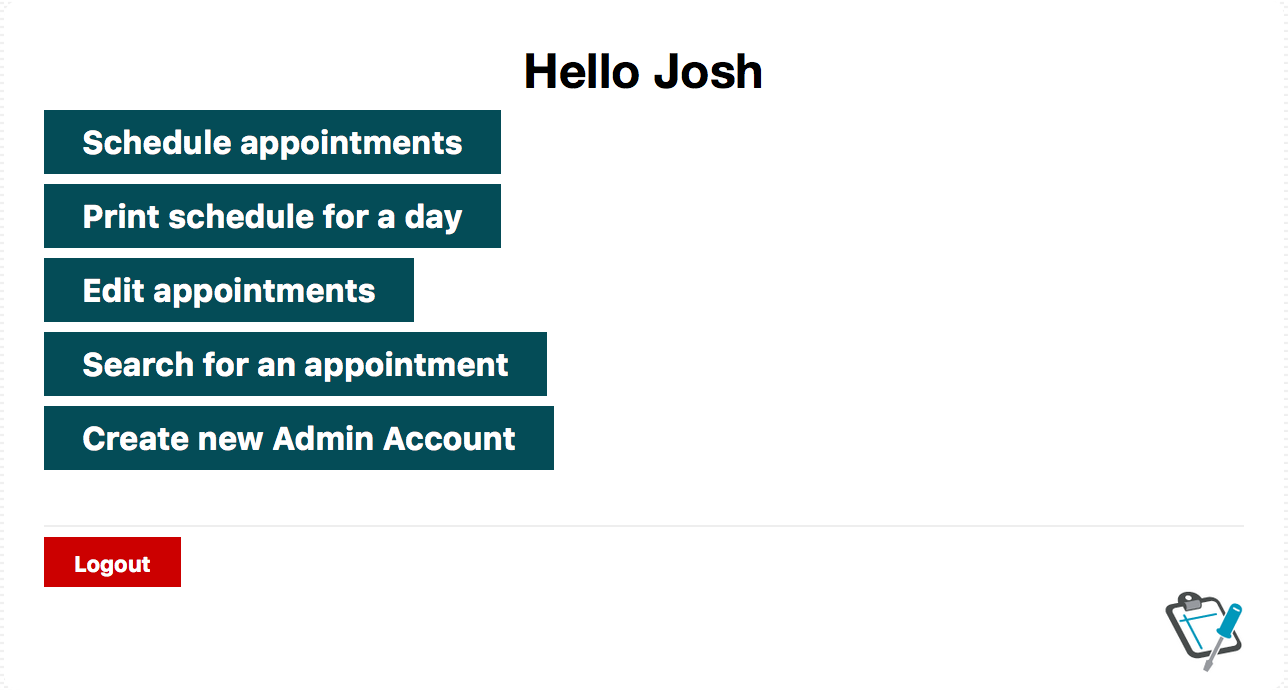
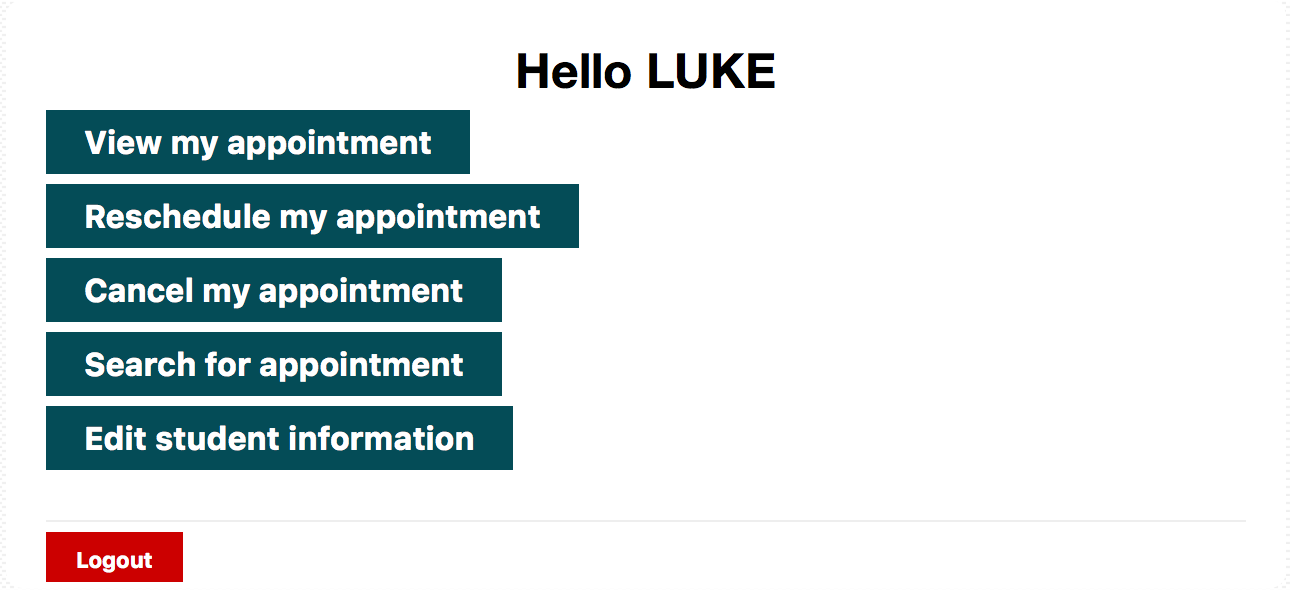
We also added CSS types for the website’s password and date fields so that they matched the CSS for text fields. This made the pages look more uniform and professional.



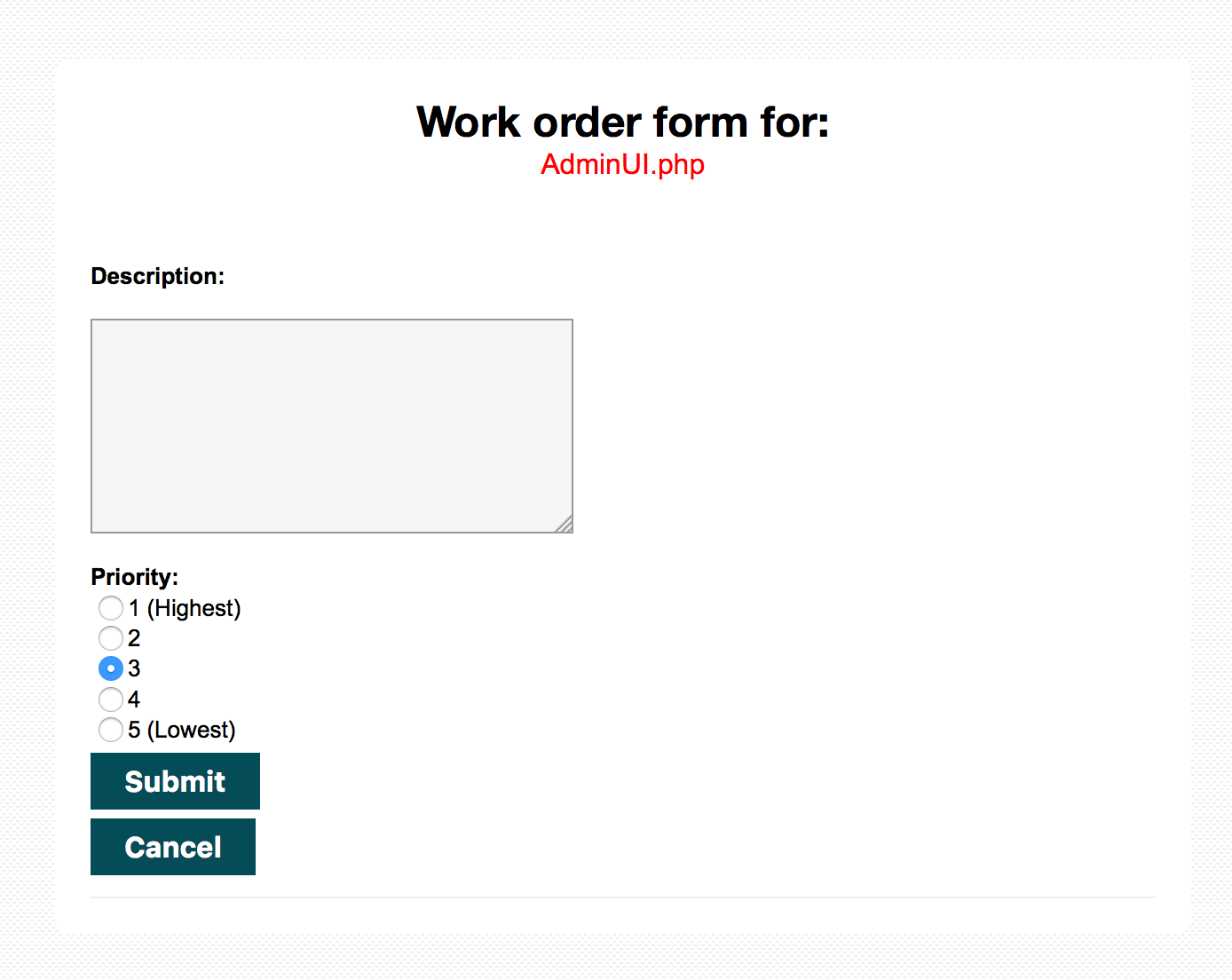
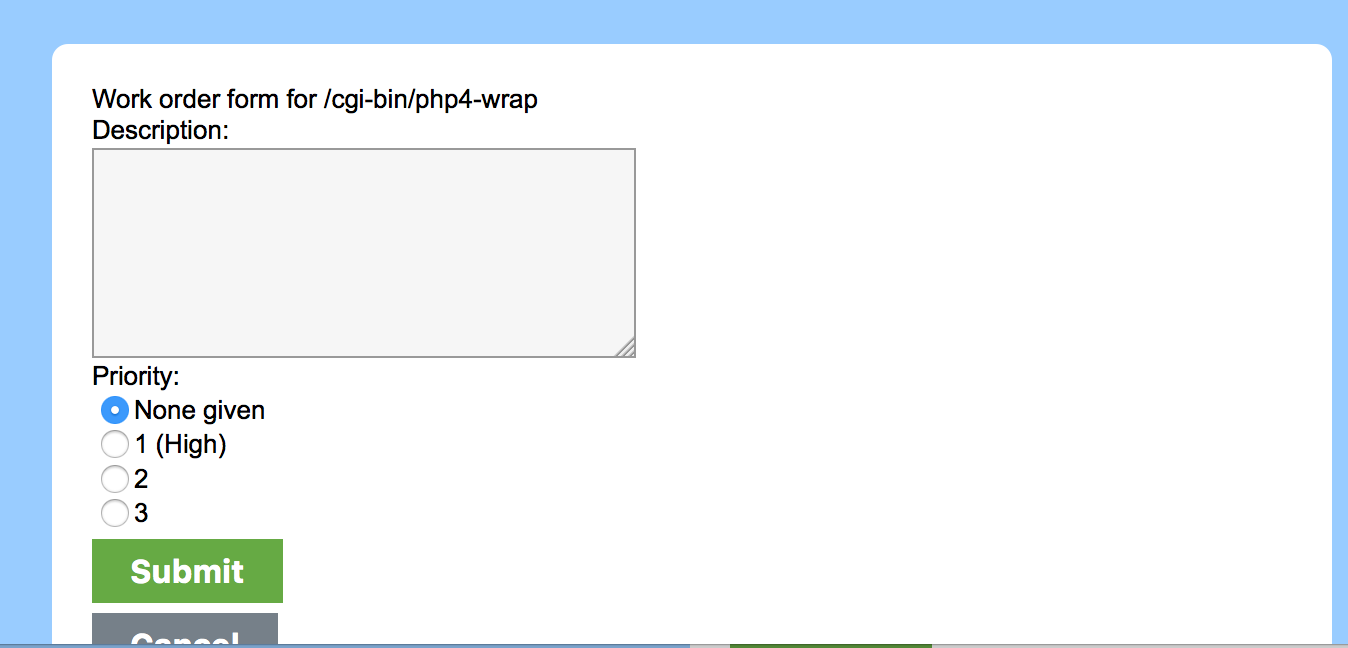
Another one of the changes we were supposed to make was to change the work order screen so that it was larger and took up more space. The CSS design was updated with the rest of the website, and the code that gets the name of the webpage where the work button was pressed was fixed. The heading for this page was centered, and the broken page was placed under it and put in red so the user knows what page they are talking about. We did away with the None given priority because the user is most likely not going to select anything different, unless it is a major problem, and we opted for a 5 possibility rank with 1 being the highest priority, and 5 being the lowest. It automatically selects a middle of the road priority that can be changed by the user. We also made sure that the window that opened showed all of the work order screen and not just a part of it.



Another part of the project was to eliminate all $\_SESSION variables except for userID. We did this by first creating a “Base” class that contained common functions that should only be created once, and then by creating “Advisor”, “Student”, and “Appointment” classes that are used to get information form Proj2Advisors, Proj2Students, and Proj2Appointments that would previously be stored in $\_SESSION variables. The constructors for these classes are created using the userID $\_SESSION variable as well as the $COMMON CommonMethods variable that is recreated on every page. We added accessor functions for easy access to data as well as so we didn’t have to retype SQL queries on every page every time we needed some kind of information, and we added a create function for each class to be used when information needs to be added to a database.

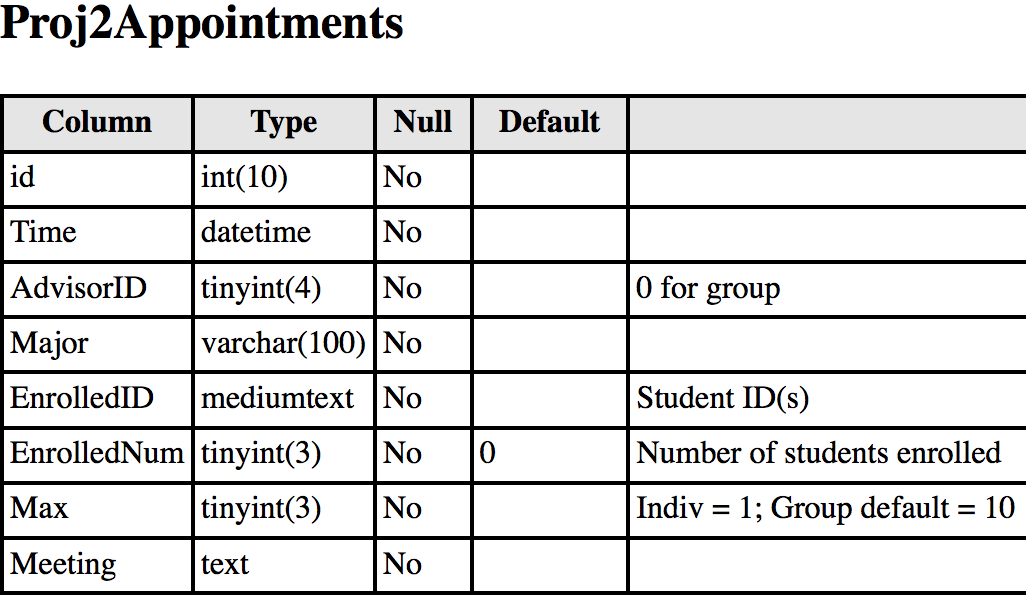
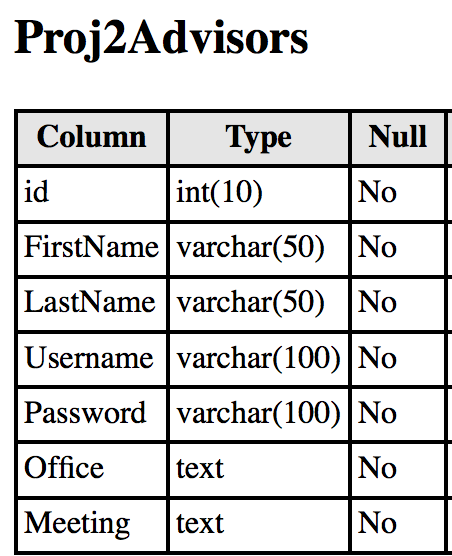


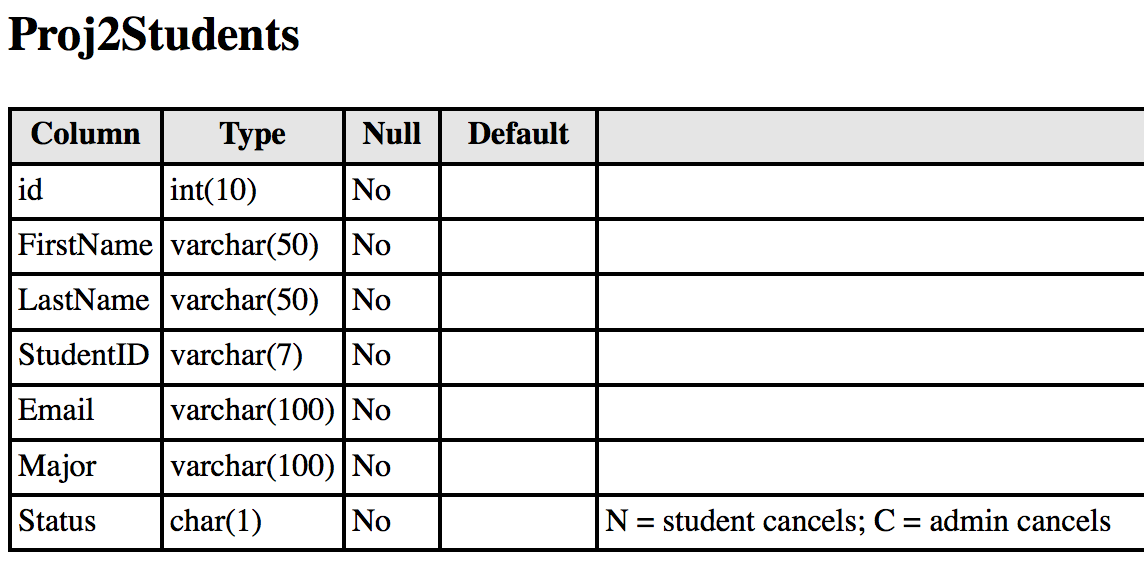
Both pages have a variety of functions. On the student side, the user can sign appointments, search for appointments, and edit your information. If you have already signed up for an appointment, then the View my appointment, Reschedule my appointment, and Cancel my appointment buttons show up and allow the user to View, Reschedule, and Cancel their appointment upon request. On the admin side, the user has the option to schedule group and individual appointments, print today’s schedule, print the schedule for a specific day, edit appointments, search for an appointment, and create a new admin account.

This project wasn’t completed without challenges. Because we eliminated the $\_SESSION variables from the project by creating new classes that would collect the information that would have been stored in $\_SESSION variables, we had to go piece by piece and debug the classes while incorporating them into the pages. Sometimes, we would find that there was an error in the classes themselves, and other times incorporating the classes’ usages into the website was just as difficult. There were times where we would have to add, remove, and change code found in the classes in order to get them to work properly.

Another challenge that we had was with the work order page. The way it was set up caused an error with the piece of code that would get the name of the page that the user was on when he/she clicked the work order button, so instead of it producing the page name, it would say “/cgi-bin/php4-wrap”. This took some research, but we found out that by changing the function from $\_SERVER[SCRIPTNAME] to basename($\_SERVER[‘PHP\_SELF’]), it would give us the proper page name.

**Part 2: Your Personal Database Table Setup**

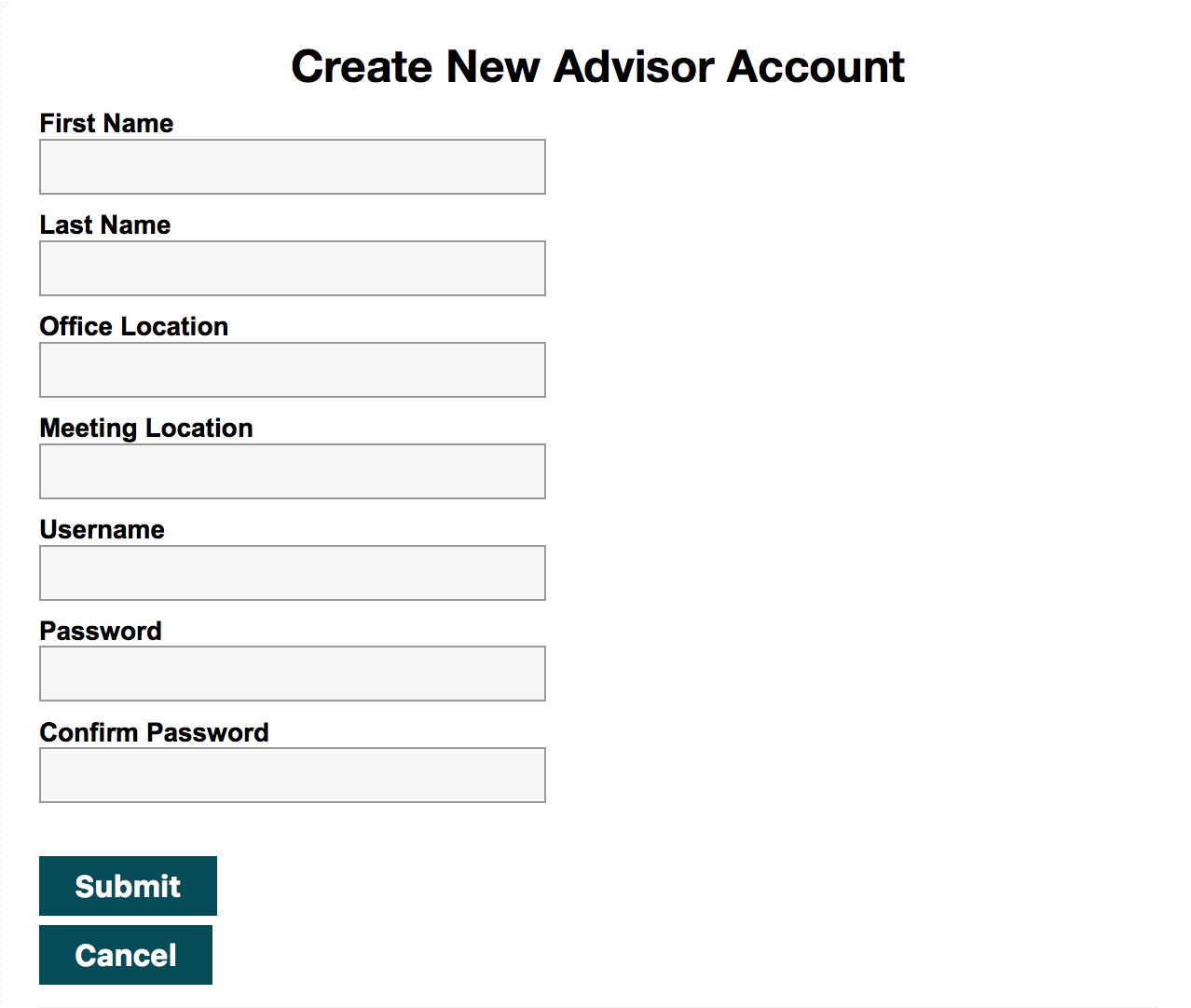
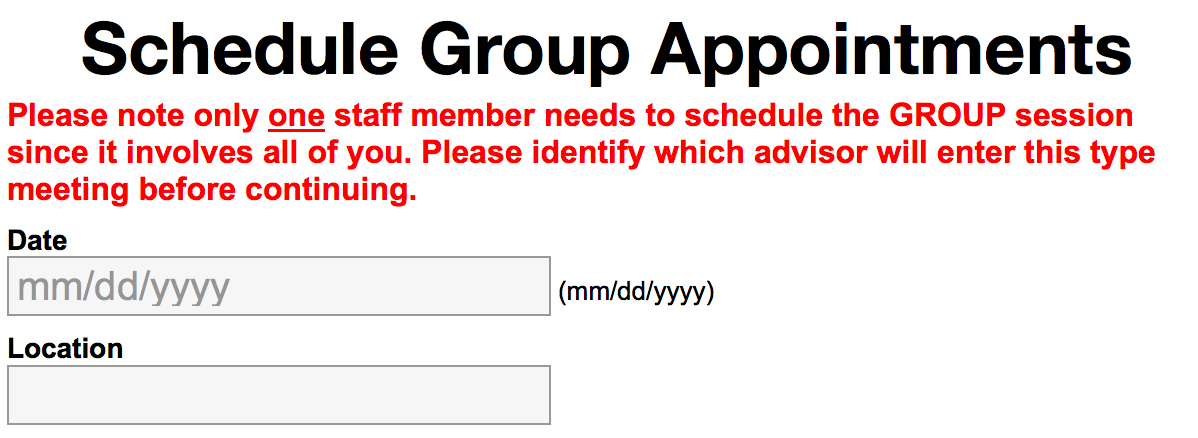
****

****

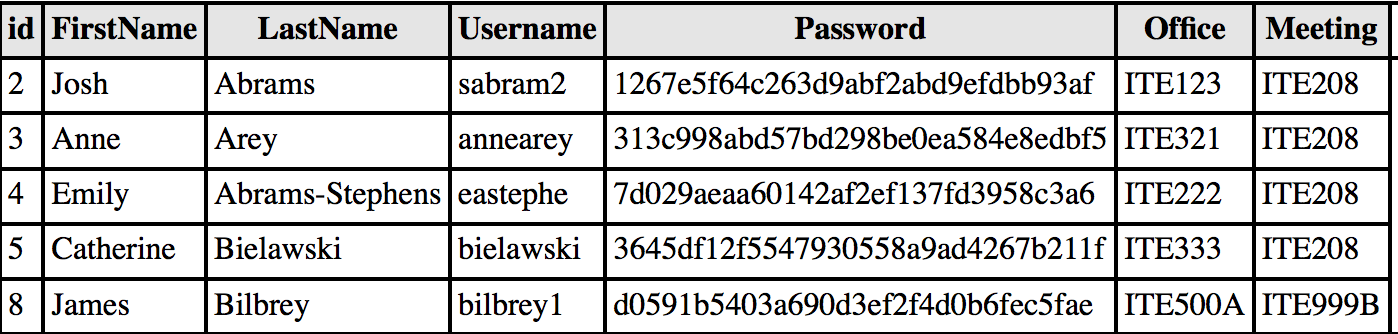
Another part of this project was to add Office and Meeting Location columns somewhere in the database so that the advisors could set their offices and where they would be meeting with students. We added Office and Meeting columns to Proj2Advisors, and then also added a Meeting column to Proj2Appointments.



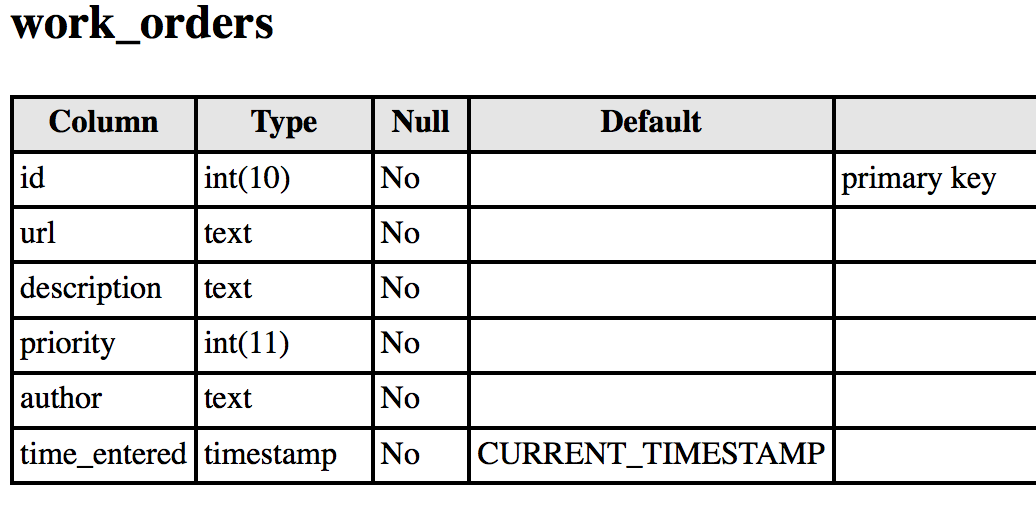
This information was also put in the View Appointment page for easy access to the students



Fields to set the meeting location were added to Schedule Group Appointments, and fields for the user to input the advisor’s office and meeting location were also added to the Create New Advisor page. The Meeting field was not added to the Schedule Individual Appointments because we added it to the Create New Advisor Account page, so the user didn’t need to input it every time he/she wanted to create a new individual appointment.

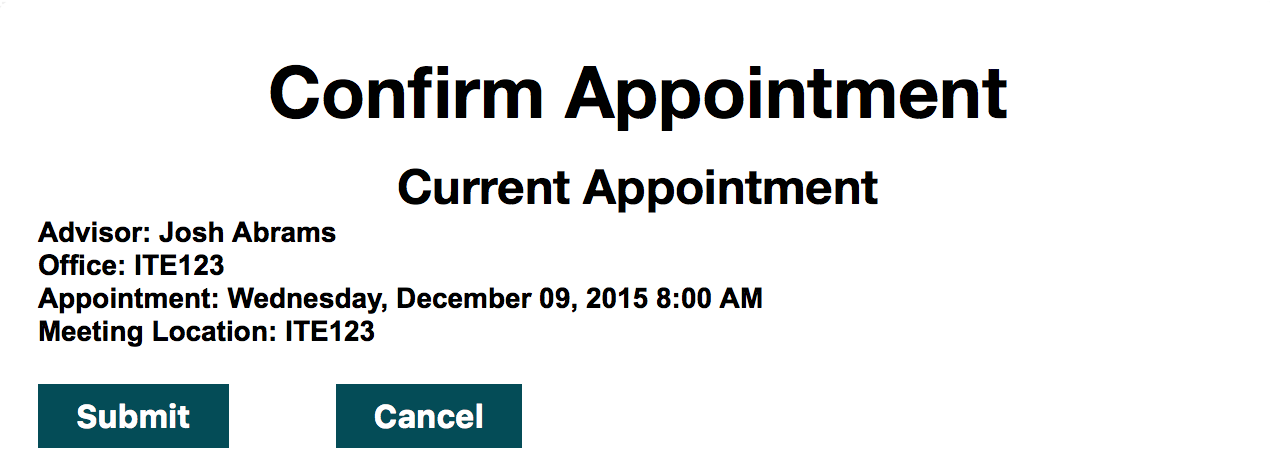
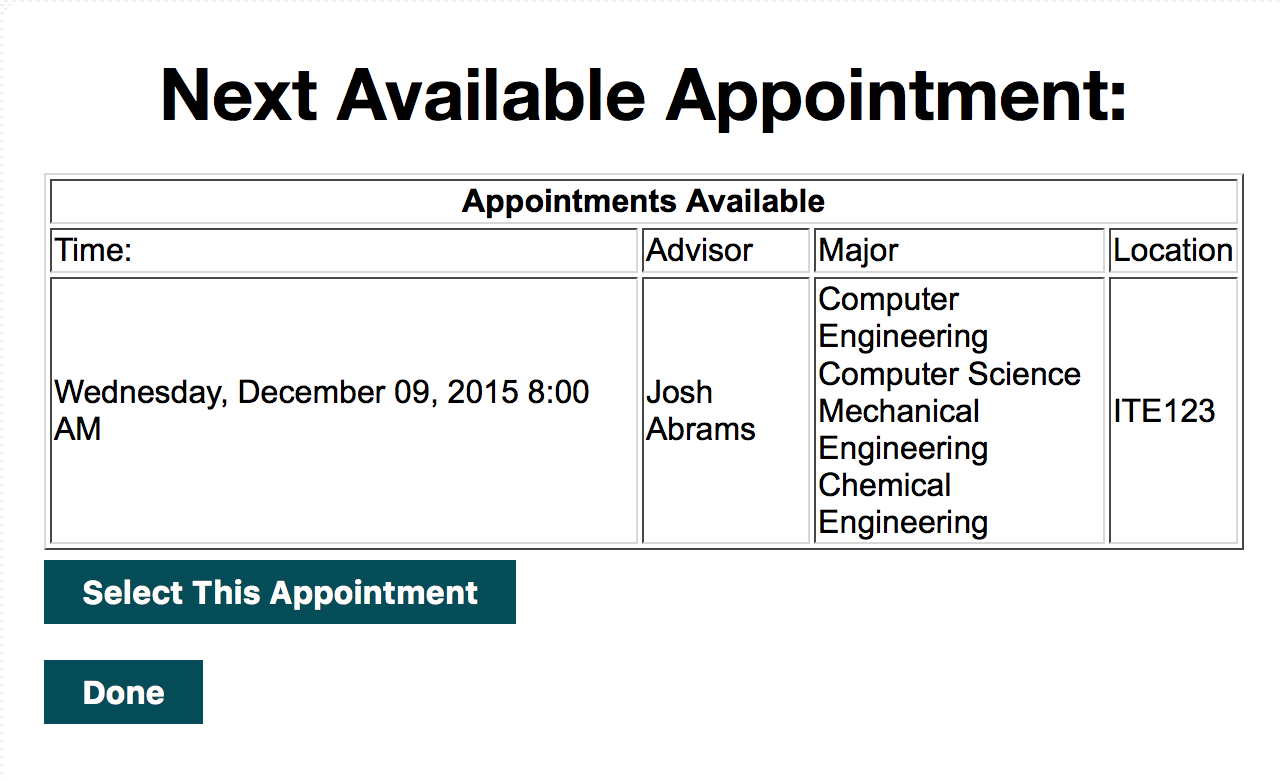


We also encrypted the Proj2Advisors using the MD5 Hashing Algorithm. This was done because we realized, in a real world application, people often use the same password for everything, so it should be guarded against people trying to steal information.

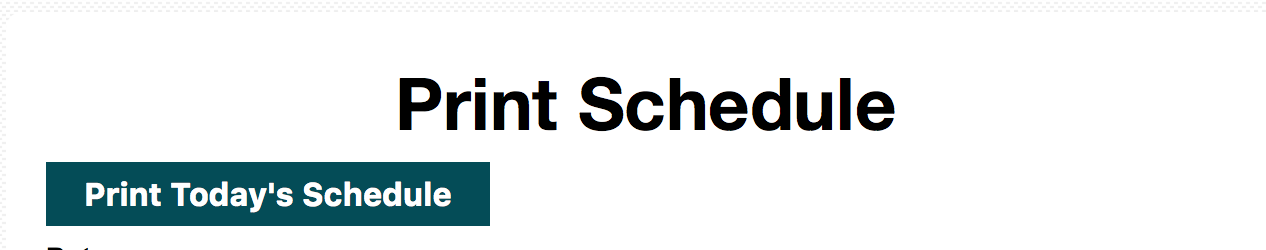
****

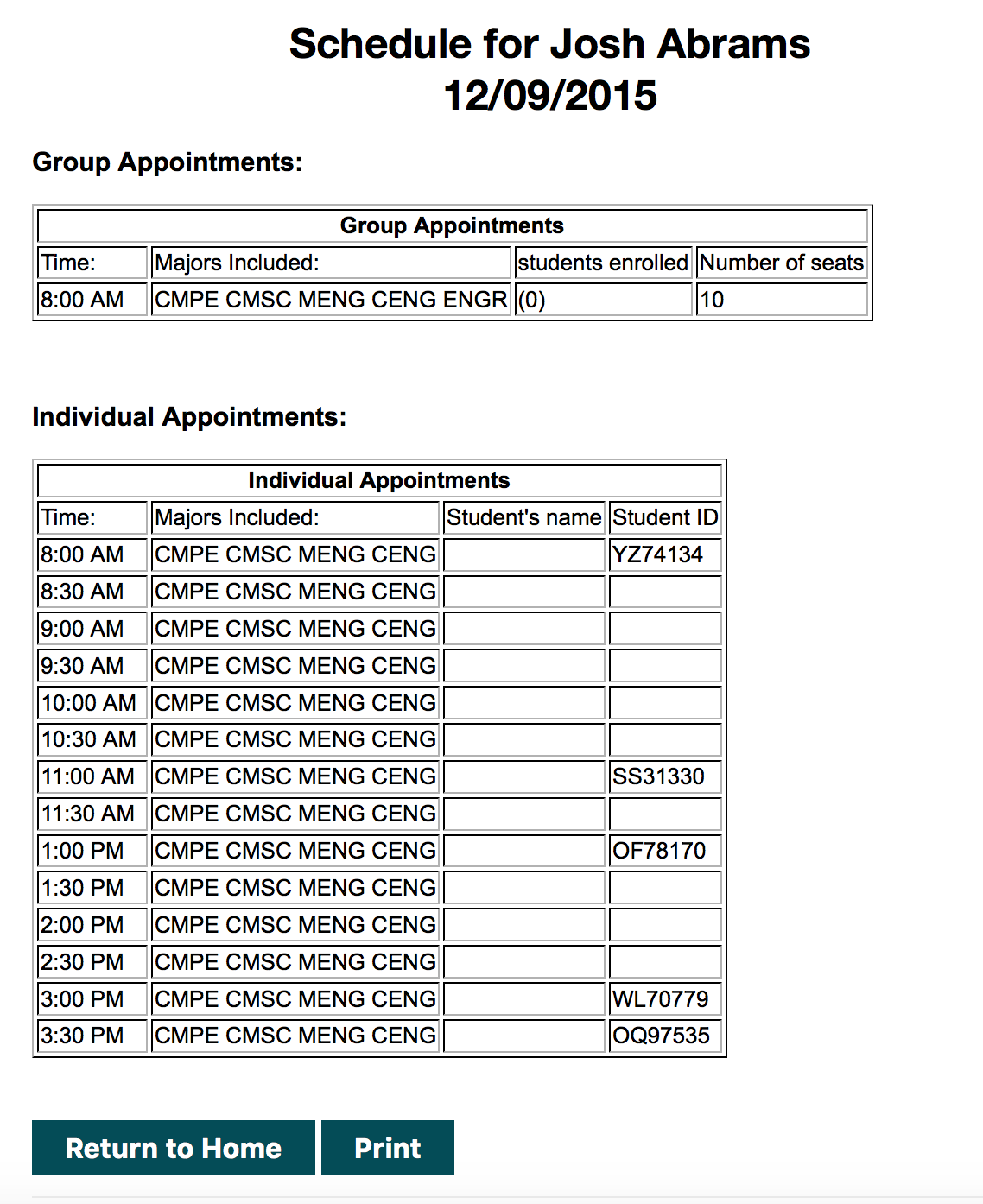
Along with changes we made to the already initialized database, we also created the work\_orders database. This database was created to store the information from the Work Order page. We realized this was needed when we were redesigning the Work Orders page. We saw that the webpage used a work\_orders database, and then created it using the information that was needed by the website.

**Part 3: Improvements**

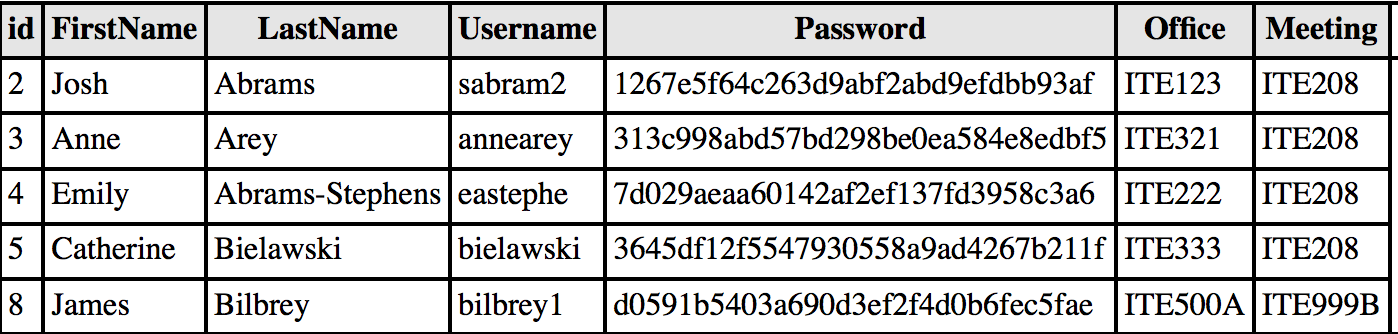
****

One of the first improvements we made was we added a Find Next Available Appointment button. What this button does is it redirects the user to the Student Find Next Appointment webpage, which searches through Proj2Appointments to find the next available appointment that contains the user and has an EnrolledNum less than the maximum amount of people. When this information is displayed, the user has the option to select this appointment to sign up for. We added this feature because we realized that most likely the user is using this button because he/she is in need of an appointment quickly and doesn’t care who it is with or what type of appointment it is.





Another improvement we made was we added a Print Today’s Schedule button. When we were testing the Admin Print Schedule pages, we realized that having a print today’s schedule button would be convenient, so we created a form that has two hidden inputs date, which is set to the current date, and type, which is set equal to Both, and uses a submit button to push the information to Print Results. This was added for user convenience. The user still has the ability to specify a date and/or time, but if they don’t want to do that, they can just get the schedule of the day. We decided against converting the Major abbreviations to their full values in order to save space in the table. We figured that when the went to print out his/her schedule, they would want a concise, easy to read page that didn’t take up too much space. This also saves paper when the user wants to print it out, which, over time, saves money.



As mentioned in the previous section, another improvement we made was that we encrypted the advisor passwords. We used the MD5 Hashing Algorithm to encrypt the admin’s password when it is placed into the database, and during login, the password is first converted into its MD5 hash value, and then compared to the database to see if the user inputted the correct password.