2014

Eastern Michigan University Honors College Application Online System

Andrew Greiner

Eastern Michigan University Honors College Application Online System

EASTERN MICHIGAN UNIVERSITY HONORS COLLEGE APPLICATION

ONLINE SYSTEM

By

Andrew Greiner

A Senior Thesis Submitted to the

Eastern Michigan University

Honors College

In Partial Fulfillment of the Requirements for Graduation

With Honors in Computer Science

Approved at Ypsilanti, Michigan, on this date 6/5/2014

I. Introduction

Historically, most forms, data, receipts, and other works were done with paper. Now most information is going electronic and moving away from paper. Using electronic versions of paper documents makes searching, adding, and distributing much faster and easier. Here at Eastern Michigan University, the Honors College still uses paper forms and applications for many things. With society shifting to electronic paperwork, the Honor's College has also decided to start shifting in order to make things easier and more efficient. This is where I came in. I assisted in converting their old paper system to a new software program that not only makes things easier, but also allows for more expansion and customization in the future.

II. Original Process

The Honors College staff and I discussed what was their current application process was. I was told that the student must first pick up or download the five page application. They then complete it obtaining any letters of recommendation they may need. Next they must then mail, scan and send as an email attachment, or bring into the Honors office the completed application and letters of support. The Honors staff checks to make sure all data matches what the school has on record, then determines if the student is accepted or rejected based off of their application. This information is added to the colleges' database. Lastly, an email is sent to the student regarding their status with the Honors College.

The entire application process takes a long time and there are many opportunities for paperwork to be lost or misplaced. Sometimes applicants send in their application and

the staff has to wait for letters of recommendation to arrive before processing can begin. This process can take weeks. For the application to work more efficiently, there needs to be a system of organization that would add all steps in working together, while tracking application progress as well as capturing downloadable statistical data. A computer program could make this process more time efficient and reliable.

III. Client's Requested Design

The first project outline was a simple web form that matched the application and stored the results. The Honors College staff did not have any extra features that they wanted inserted into the site. They wanted a direct copy of the application packet converted to an online version. They didn't really have an idea of what happens to the data after it was filled out. In a nutshell, they wanted me to take the student's data and put it in their Filemaker database.

After creating the first prototype, we realized that we could not just import data into the database. Also, there needed to be a way to hold the data until the staff had reviewed the application and decided if the student was accepted or not. We talked about storage using spreadsheets or emails; which would not have offered any significant improvements. It was finally decided to place all data into a database which I would create. This way, they could get a spreadsheet of all the students who applied and move the accepted students into their own database tracking system.

At our next meeting, I demonstrated my current design which was very basic with some search features. At this point, I believe that the staff realized the potential of a computer program. They started to see that what they did by hand could be done

automatically. This project which had started as a small simple idea was now growing into a larger concept.

Knowing part of the capabilities, they wanted more. They wanted the student data to be stored forever and wanted to run queries on past data, new data, and on all data. The staff wanted a way to print off the student's application if need be. The staff also wanted a way to see and manipulate the data in a spreadsheet.

Every meeting after the previous meetings was to show the results from the last meeting, and for the client to ask if more things could be done. The Honors staff had to create emails for every student that applied to the Honors College. If a student had qualifying grades, a good recommendation, and was in good standing with the college, they would send them a certain email. If a student was on the opposite end of the spectrum, the student would receive a rejection letter. This email process was one which took a lot of time. The staff wondered if there was a way to make that aspect of the application process easier.

While the staff had ideas of what they wanted, I knew of some other things which they would need. I knew they were going to need some administrate tools for their employees. From my computer background, I know you always need back-end tools for the users. I spoke to them about creating a simple administrator add/delete system which would allow them to have more than one username and password. This is something that I think almost every system should have.

Next, we came to the issue of letters of recommendations. The Honors College's old system was to either receive them with the whole application all at once, or receive them all at separate times. During the period of times between receiving the application

and receiving the letters of recommendations, papers may get lost or forgotten about. We needed to plan a solution to fix this problem.

Our first idea was to have our email system send out an email to the recommenders to ask for a letter about the student. This would cause a lot of emails to be sent to only one email address. The person in charge of the email address would have a hard time wading through the emails and getting things organized. That was not the goal of this project.

After some time, we figured out a new solution. We wanted a way where the recommender could send the letter directly to us electronically, just not through email. The staff decided on having a location on the site where the recommender could log in and upload their letter. This would match the letter with the student's application without the need of an employee matching the papers.

The Honors College staff also saw an opportunity to receive more feedback from the recommender. They wanted point system questions to be asked about the student. They wanted something more than just a letter of recommendation.

Finally, after time had run out to add more features, the polishing of the application began. They explained how they wanted the wording on different parts of the application and that they wanted validation checks when the application came in. They wanted things to look and feel smooth for both the student, the recommender(s), and administrator.

IV. Design

Creating this project was a learning experience for me. I had only handled some basic HTML, CSS, and JavaScript. I did know some SQL, but I had never used PHP. This whole project was me learning on the fly.

The staff at the Honors College wanted software that they could use on any computer. Because of this, we decided to go with a web based software. Like most web based software, I knew the pages needed an HTML skeleton. The main title of each page was to be created using HTML header tags and the main content container was created using HTML div's.

A style sheet would also need to be created. A page without any CSS seems cluttered and it is difficult to read. I wanted to make the pages look similar to the current application. The style sheet had to be constant with every page. I figured if there were multiple style sheets, it would make the site look less cohesive.

I decided to go with a MYSQL database to house all the data. Since the idea was to make the student records mesh with their main database, I created a big table just like theirs. I also needed to create a few small tables to hold different user login credentials.

Since we were going with a web based software and using a database, I needed to do most of my coding in PHP. Eastern Michigan asked that I put the database variables in a separate file to keep logins to their database safe. For every page that needed the database, I would have to reference it to the database variables file.

To make sure that incoming data from the students was somewhat good data, I would need JavaScript. This language would be used to check each input field to make sure that it was filled out and in the correct format, before sending it to be processed.

With some of the site, there would be content that would not need to be shown. For example, the student application has three sections, first year students, transfer students, and current students. A first year student would not have to fill out the current student section of the application so I would need to use JavaScript to show and or hide the needed divs.

The big picture of the site was to have three different parts. The first part is the student's side. This part needed an HTML form page to receive student data, and a PHP and SQL page to put the data into the database. The second part is the Honors College staff's side. This part mainly needs SQL and PHP. This part needs to use a lot of pulling, manipulating, and pushing data to the database. The third part is the recommenders. This part uses HTML to allow the recommender to write the letter and PHP and SQL to send that letter to the database.

V. Implementation

I first created a basic web form using HTML that allowed the user to input data. I decided to make the look of the site somewhat match the look of the application. For this process, I created a style sheet and wrote some CSS code to organize the form in a presentable way. While I was creating the form, I ran into the problem of different types of students. Each type of student, first year, transfer, and current, had a different page with different questions to be answered. From this point, I decided to insert some JavaScript code to make the divs show after the type of student was chosen. When a radio button is clicked, it runs the JavaScript function that hides all non-related divs, and shows the related div. At the end of the form, I added a submit button to send the results

to the next page. I then realized that I must validate my form. I created a JavaScript function which checks every form item to make sure that all the data is there and in the correct format. I also implemented a way so that the function knows what fields should be empty because of what type of student the user was.

Next, I needed to learn what happens after the form is submitted. I knew it had something to do with PHP, so I explored some online resources on PHP. W3Schools has nice tutorials on how to use and understand PHP. I also needed to learn how to import my data into Filemaker. After browsing a few web forms, I had an idea of how it could be implemented.

We decided to go with a database that I would create with MySQL. This database would house all student applications that were submitted. Since most of the staff does not know SQL, I also needed to create a user interface for them.

I decided to create the table to match the current table as best I could. This table was one large table with many records, which is how I based my table. After creating the table, I created the PHP page, which would take the input data and place it into the database. I found that you could use POST variables to grab the data that passed from the previous page. This made handling and manipulating the data much easier.

After much PHP coding, I found my program wasn't doing anything. I found out that if you want to use PHP, you must put this on a server. At the time, I didn't have a server, so I decided to make my laptop a server. I ran Apache on my laptop that ran my pages and my database. The Apache software allowed my site to be hosted locally.

Now that I had a way to store the data somewhere, it was time to create a user-friendly end for the staff. We decided that the best approach would be to go with a website rather than installed software.

I first needed to create a login page for the staff. We didn't want just anybody to get into the database and obtain student information. The login page was a simple login, which asks for a username and password. Rather than having hard coded usernames and passwords, I needed to create a table which held all the administrator usernames and their passwords. For such a simple program, I didn't create a way to encrypt every password.

After the login page, I created a simple PHP file which grabs all of the student records in the database and displays them. I found out that someone could just type in the site address and just skip the login screen. I went online and found out that I could use SESSION variables that would make sure that someone was logged in. SESSION variables are variables that are stored in the browser and can be retrieved from any page. I created a variable that was set to "1" if the user had logged in. When the user logged out, or the browser was closed, the SESSION variable would be destroyed. So now, on every page I needed someone to be logged into, I added at the top, "if($_SESSION['view']!=1) { /*Go back to login page*/ }".

Next, I added a search function. We knew that once the table started filling up, it would be hard to find the record they wanted. To achieve this, I needed to create a somewhat dynamic query creator. I created a page that had radio buttons for the current status of the student, textboxes for specific names, and drop down menus for different sorting. When the user hits submit, it creates a SQL statement from all the data that had been input. The statement started out with, "SELECT * FROM (table_name)". I then had

various "if else" statements to check whether it was the first condition to add "WHERE", if the data field was empty, or if it was a more complicated search (needing ORs and ANDs). After this statement was created, I execute it, and then display resulting in a HTML table.

After viewing this new search results page, since we could see all the results, we wanted to be able to manipulate the data. I then decided to add a checkbox next to every record that was found and a "select all" checkbox at the bottom. With this, the user is able to select all the students that they want to manipulate in the database.

The first function I added to the results page was the change status option. The user could select students and change their status to 'accepted', 'not accepted', or 'pending'. After the user choose the students and the new status, they would hit submit and reload the page. I used a POST array variable to retrieve the checkboxed students EID. With this data, I do simple SQL queries to update their status with the new chosen status in the database.

I created an administrator tools page off of the administrator search dash. I added a drop down menu on the new page, which allowed the administrator to choose what they wanted to do. Since I wanted to make a dynamic email message system, I created an option to change the email message.

We knew that we wanted a message for acceptance, not accepted, and new applicants. Rather than hardcoding these messages, I created a database table that held these messages. Each email record in the database has four fields, the email name, the subject, the message, and if it was active or not. Since the message could be very large, I

changed the type from a varchar to a blob. This allows the message not to get cut off if it was too large.

For the page, I used text areas, text boxes, and a radio button for the user to update the email message. The user first selects which email message they wish to update. Next, I do a query that grabs the record for that email, and populate my input boxes with the current email details. The user can now modify the current email message and then press submit to save the changes. I realized that this message was going to be sent off to many different students. I thought it would be a good idea to add variables for the user to place in the email. For example, I added the variable that they would type: *fName*. When the message is getting ready to send to the student, it would update anywhere it found that string of characters and replace it with the current student's first name. I did this same thing with a few other variables they might want to be added to the email like EID, date, and last name. I used PHP's function, str_replace() to find and replace the variables that I wanted.

While testing this email update page, I kept getting weird errors or things cut off. After a bit, I realized my mistake. When creating SQL queries, you use apostrophes to indicate where the string starts and ends. So anytime I wrote words like I'm or can't, it would cause this error. I decided to create a JavaScript function that would grab every user-inputted apostrophe (and quotes) and places a slash in front of it. This changes the apostrophe from a literal to the value of it. The function then updates the user-inputted strings and reloads the page, causing the email to be updated.

I went back to put some more functions that an administrator may use later. I updated my adminTools.php page with more drop down menu options. These mainly

dealt with administrator logins. I added an add new administrator page. This page allows the current administrator to create a username and password for a new administrator. Since the site had an add administrator now, it needed a delete administrator. This new page queries all the administrator usernames and allows the administrator to choose which one they would like to remove before any administrator removing could be done. I created two input boxes that ask for the current administrator to type in their password twice. This way, it reduces the chances of accidently deleting someone that you did not want to delete. I hardcoded one administrator username that could not be deleted so the site does not become useless. Finally, I added a change password page. This page asks for your password, and the new password twice. I thought it would be best if you could only change your own password. If you can't remember your password, you can have someone delete you and remake you.

Next, we decided to go with a generate report section of this site. I figured that there was some way to export the database table to a spreadsheet. I first ran the SQL query, "SELECT column_name FROM information_schema.columns WHERE table_name = '(table_name)'". This query grabs every field name from the table. I then made a simple function that takes all the field names and places commas in between them. I next created a SQL statement from all the EIDs that were passed in. I used fopen() to create an outgoing csv file. I included my field names in the csv file using fputcsv(). Finally, I ran my EIDs and pushed all the results into the csv with a while loop. Now, when a user selects generate report, a download of a csv file pops up for them.

After some more testing, I got annoyed by constantly deleting students through SQL statements. I figured the Honors College might need to delete student function

sometime. I created a page which took in the EIDs of the selected students that were to be deleted. This page has a huge warning that basically says, "are you sure you want to do this?", and displays all the students that will be permanently deleted. I put two password boxes that the user has to fill in before they could delete them.

The next thing to be added was a way for the Honors College to print off a student's application. This function was where I spent a lot of my time.

Online, I found a PHP library which is used to write on pdf files. I had the application pdf which the College uses to print for the students. This made it easier then creating a new pdf look. For this page, I included the pdf writer files, "fpdf.php" and "fpdi.php". These includes allowed me to write to a pdf file. The basic way this works is by first opening the pdf and setting it as the source file ($pdf->setSourceFile('app.pdf')). Next, you import the page number you want to work on ($pdf->inportPage(1)) and use it as a template ($pdf->useTemplate()). Once that was set up, I pulled all the needed data from the student and starting placing them on the page ($pdf->SetXY(); $pdf->Write();). The only way to place them in the correct spots was to use integer numbers for every location. With over thirty plus data points to place, setting each one up took some time.

We had some trouble figuring out what to do with the student's letters of reference. We wanted something that was more automated and easier to manage. Having the application electronic and the letters done by hand wasn't smooth enough and would demand too much hands on work for the Honors College staff to keep everything up to date. We decided the best approach would be for a whole new sub site for the references.

To do this, I needed to create a login page for referrers. I used the same basic template as I did with the administrator login page. With this setup, I needed a way to create and distribute the reference's username and password.

I created a spot on the application for the student to insert the name and email of whom they are getting the letter of recommendation from. I also created a new table, which housed a recommender's username, password, and which student they are writing about. After the student hits submit on their application, I grab the reference's email and chop off the "@" and what follows. This becomes the recommender's username. I then tested this username with the database. If the username is already there, I increment a number at the end of their username until I find one that does not match. For their password, I randomly generate a string of six letters and numbers. I then put this information into my already made mail system and send the important data to them.

Once the recommender receives the email, they can now login to the link I gave them in their email. After they log in, I created a text blob area for them to write their letter. After they finish that, they are asked a few questions about the student. Finally, the recommender hits submit, and their inputted data gets sent to the student's record. The recommender's login credentials are removed to save space in the database.

I now had a way where everything the student needed to apply became automated. Now, I had to create some sort of alert system, letting the Honors College administrator user know that a student had all his/her information ready. I created a new status for the student, showing the administrator that they were ready. Before, I had 'pending', 'accepted', and 'not accepted', now I added 'ready'.

With the new recommendation system setup, we needed a way to look at the letter. I created a page of the administrator dash that displayed the information; this page was pretty basic. All the page does is pull in the recommender's letter and questions to a page and displays them.

After all these features were added to the project, I added a few more tweaks here and there to make things run more smoothly. I cleaned up a lot more bugs, and did a lot more testing. This project is functional and is ready to be placed out on the web. This project is definitely far from being finished however.

VI. The Final Product

The initial project started out as a three page site; the application, the data side, and the style sheet. The ending site count was now at two hundred and sixty five files. Most of these files are libraries that are used for the site but thirty four files I created. This final product had grown significantly since it started, and it is not even close to being done. There are a lot extra features, add-ons, and other miscellaneous parts that the Honors College would like to have added.

The final product consists of three parts, the student side, the Honors College side, and the recommender's side. All of these have their own separate parts, but all are working together.

The student's side of the site is the application page. This page allows them to input all the basic information, name, EID, address, and so forth. It also allows them to add their essay that they need to write to become an Honors College member. Additionally, it provides a spot to email their recommenders for a letter of

recommendation. This site allows the student to submit their application without ever leaving their home.

The recommender's side of the site allows them to send their letter to the Honors College without having to seal an envelope and give it to the student. The recommender has a unique username which allows them to log in for a specific student, write their letter of recommendation and also rate them on a few questions. All of this is also done without them leaving their own homes.

The Honors College side of the site allows the staff to manipulate and view the applications in a simple way. The staff is allowed to view students through a variety of different search credentials. It also allows the staff to print students' applications and create reports from students. Emails are now sent automatically for the staff. They do have the option to dynamically change the messages for each one. The site allows the staff to do all the work of accepting or rejecting students with a few clicks of the mouse. There is also an easy way to change the questions that are asked about the students to their recommenders.

There is additionally an administrator tools side to the site. This part of the site allows the Honors College to handle administrators on their own side. They can add, delete, or change the password of administrators. I tried to create this site to be dynamic so that more features could be added easily and quickly.

VII. Possible Problems in the Future

With all software, there are always bugs. Any programmer will tell you that they know how they could break their own software. The hardest part about this project was to make sure that no one else would be able to.

I have regulated a bunch of special characters in the application portion of the website. I have accounted for a majority of well-known issues that generally arise from a majority of users. I have not accounted for some names that have special characters in them. When doing SQL calls in PHP, the values are separated by apostrophes and commas. If some names or words contain apostrophes or commas, it could throw off some of the new data being passed in.

I have also accounted for some of the data checking. For example, the EID must start with an 'E' or 'e' followed by eight numbers. I do not check to make sure that phone numbers are of correct length or contain no letters. I do not check to make sure that zip codes are a certain length or contain no letters. Making sure good data that is coming in is a big process, and I think should always be worked on.

Since the application part of the site was the first part of the site that I worked on, and it has been changed so much, it is not very orthogonal. One minor change on the application page may throw off the page which puts the data into the database. Since the student table is huge (first created to match the Filemaker's table) and will only have one third of the data filled out (the student has different questions whether they are a first year student, transfer student, or a current student), the data input into the database is very picky with the order. If the next programmer has time, I would recommend breaking up

the page where the data is input into the database into three sections. This will make changing of parts on the student's side much simpler and quicker.

For the Honors College site, I see a bunch of small problems that might happen. The database for student data is not backed up. If an administrator accidently changes every student's record to accepted, there is no way of going back. If an administrator deletes all the students, there is no way to revert these changes. If I had time and the space, I would put up a backup database which allowed the administrators to revert changes which were made. Also, just like the student's side of the site, special characters could through off the SQL queries created through PHP. If an administrator decided to make their username "I'mAnAdmin", I don't know the results of what may happen.

For some features, I realize that I grab the data being passed in for single student features, I either grab the first or last student chosen. When grabbing data from a database using PHP, you must use a while loop to grab the data. If I do not add a break statement in the first loop, it will grab that last student selected. If I had time, I would like to create an alert that would not allow the user to select multiple students for that feature.

There are also some problems that may arise from new programmers working on this site. A lot of my code rests on the naming convention of variables that I used. If a programmer changes the way something is spelled, or even making something all lowercase, it may break something else in the site. I have never worked on something this big before and I think it would be a wise idea to create a variable naming/usage manual for future programmers.

Because of lack of time, I have not been able to test my software well enough. I have tested this site a lot using my own local host laptop to fix things. I have noticed that

features that work on my own machine, do not work on the cmich.edu site. For example, I am not allowed to query for the column names of my table. For my generated reports page, as of now it queries for all the column names and places them at the top of the spreadsheet. I had to change it to hard coded column names to be output to the spreadsheet. I have not been able to do enough testing to feel comfortable with the software being on a new host. Also, I have only tested this site with Chrome and Mozilla. I have not done any testing using Internet Explorer, Safari, or and other web browser.

I have also learned a few weeks before my time had ran out that my email system only works on simple email systems. My system will send out emails to almost any type of address except for gmail.com and yahoo.com. I am currently working on trying to fix this with a PHP library called PHPmailer. This library should work, but it is just a matter of finding time to figure out how it works and implementing it. This system should be working before the 2014 fall semester.

VIII. Conclusion

Overall, I tried to make this project simple, yet dynamic. This project has many more options that can be added to it. More features and functions can be added to make the Honors College more automated. I tried to make this a nice core system which can be added on to. I would have loved to see what this project would have been if I had been working on it since I started at Eastern Michigan University.

This project has also really helped me with other obstacles in my life. For my COSC 481 class, I needed to create a site in PHP and connect it with a database. Without the knowledge I learned from this project, I would have been far behind. It has taught me

a good path to take when designing the website and database. This project has also helped me land a job in my degree field. I had a lot of real world experience creating this project which I talked about during the interview. I felt a lot more comfortable and experienced because of this project. I do also believe that I will use this knowledge that I gained to further my work in the future. I've learned that if there is an idea, there's probably a way. If you don't know how it's done, research it and brute force your way through it. I have learned that it's good to code something you don't know. After I created a function I didn't know how to do, I was able to look at it and refactor it into something better, more efficient. I will take these skills with me where ever I go to create something. I won't let an idea that I don't know how to do stop me from achieving something that I want to create.

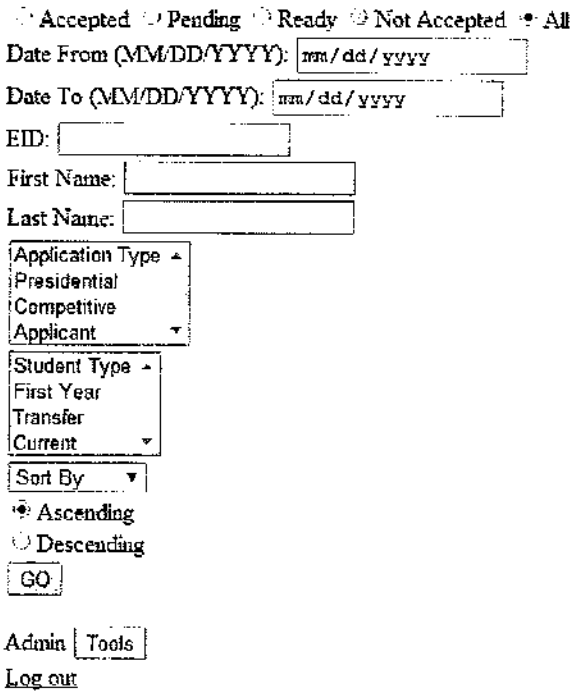
Honors Emich.edu Admissions Administrator User Manual

Login:

- First, go to the location of the login page. (www.emich.edu/honors/admissions/loginPage).
- Type in your username and password into text boxes. (Default Administrator username is EMUHonors)
- Hit submit to take you to the adminDash.php page.

Admissions Page (adminDash.php)

Eastern Michigan University Honors College Admission Page



- This page is to search for students in the database.
- At the top of the page are radio buttons to search by the student's current status.

  Accepted - Student has applied and has already been accepted into the Honors College.

  Pending - Student has applied to join the Honors College and is waiting for letters of recommendation.

  Ready - Student has applied to join the Honors College and has all the needed paperwork ready to be checked.

  Not Accepted - Student has applied and has already been denied.

  All - Searches for all students, regardless of current status.
- Date From and Date To - These inputs are to search dates from the time in which

students have submitted an application. This takes the input format of mm/dd/yyy (Ex: 03/16/2013). To search for students who applied between the dates of March 3rd, 2014 to April 3rd, 2014, set Date From to: 03/03/2014 and Date To to: 04/03/2014. If you want search for all times, just leave them both blank.

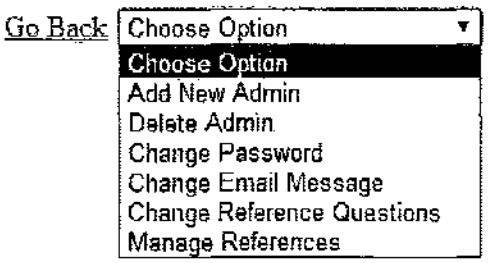
- EID, Firstname, and Lastname - These inputs search for students that match what is entered. It also searches for partial strings of characters (Ex: First Name: "drew" will return any first name that contains "drew" in it. Andrew, Drew, Drewster, etc). Leave textbox blank to not search by these fields.
- Application Type - This list searches for the submitted application type (Presidential, Competitive, and Applicant). This list allows multiple selections to search for. To use, you can highlight multiple records by holding the left mouse button and drag over the ones you want, or you can hold Ctrl and click the ones you want to search for.
- Student Type - Same as Application Type, but is for searching for the type of student they are (First Year, Transfer, and Current).
- Sort By - This drop down list controls how the searched records are sorted. You may sort by EID, First name, Last name, or Date.
- Ascending or Descending - These radio buttons go with the Sort By drop down menu. These buttons choose which way you would like the results to be sorted.
- GO - This button should be pressed after you have decided your search preferences. This button takes you to the next page that displays the results (adminSearch.php).
- Admin Tools - This button takes you to the Administrator Tools page of the site.

See Administrator Tools (adminTools.php) for more info.

- Log out - This link logs you out of the current session on this site.

Administrator Tools (adminTools.php)

Eastern Michigan University Honors College Admin Tools



- This part of the site is for the administrator settings and other miscellaneous settings.
- Go Back - This link takes you back to the Admissions Page (adminDash.php)
- Choose Option - This drop down menu is for adding a new administrator, deleting an administrator, changing your password, changing an email message, changing reference questions, or managing the recommender's login database.
- Add New Administrator - This selection allows the user to create a new administrator. To create a new administrator, enter a username and a password you would like for the new administrator. Retype the password and then click Submit.
- Delete Administrator - This selection allows the user to delete an administrator. To delete an administrator, choose the administrator you would like to delete from the drop down menu. Next, type in your password, then retype it again. Then, hit

Submit. (Note, EMUHonors cannot be deleted).

- Change Password - This selection allows the user to change their password. To change your password, enter your old password. Next, enter your new password, then enter it again. Finally, click Submit.
- Change Email Message - This selection allows the user to change the settings on an auto email message. To start updating an email, first choose the message from the drop down menu. Next, click Submit. This will now take you to the Update Email page (updateEmail.php). See Update Email (updateEmail.php) for more details.
- Change Reference Questions - This selection allows the user to change the settings of the reference questions. To use this feature, just click Submit. It will take you to the Update Recommendation Questions page (updateQues.php). See Update Recommendation Questions (updateQues.php) for more details.
- Manage References - This selection allows the user to manipulate the recommender's login database. To use this feature, just click Submit. It will take you to the Reference Database Manager page (refDB.php). See Reference Database Manager (refDB.php) for more details.

Update Email (updateEmail.php)

Eastern Michigan University Honors College Update Email

(Email Help)
Go Back
Subject:
the subject
Message:
Type the message here

- This part of the site is for changing the settings of the auto emails.
- The email that is to be changed is passed in before this page is loaded from a drop down menu.
- The Email Help link on the page shows all the variable names you can put into the subject and the message. For example, if you would like to have the student's first name be on the email, you would insert the string, "*fname" (Administrator's side: "Congratulations *fName!". Student's side: "Congratulations Billy!").
- The Subject text box is for creating the subject line for an email.
- The Message text box is for creating the message that gets sent in the email.
- The Active radio buttons are for turning on or off the auto email function. If you do not want emails to go out automatically for this message, you would set Active to "No".
- When you are finished updating the email, click the Submit button at the bottom of the page.

Update Recommendation Questions (updateQues.php)

Eastern Michigan University Honors College
Update Recommendation Questions

Go Back

Question 1:

question 1

Is active?
Yes
No

Question 2:

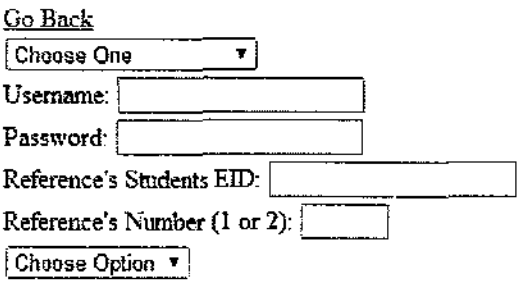
question 2

Is active?
Yes
No

- This part of the site is for updating the questions that are asked about the student to the recommender.
- Five questions are allowed to be asked about the student.
- The text box under each question is the actual text that will be asked to the recommender.
- These questions have radio buttons that the recommender can choose (1-5 and NA).
- The Is Active radio buttons are for turning on or off the question to be asked.
- When finished, click the Submit button at the bottom of the page.

Reference Database Manager (refDB.php)

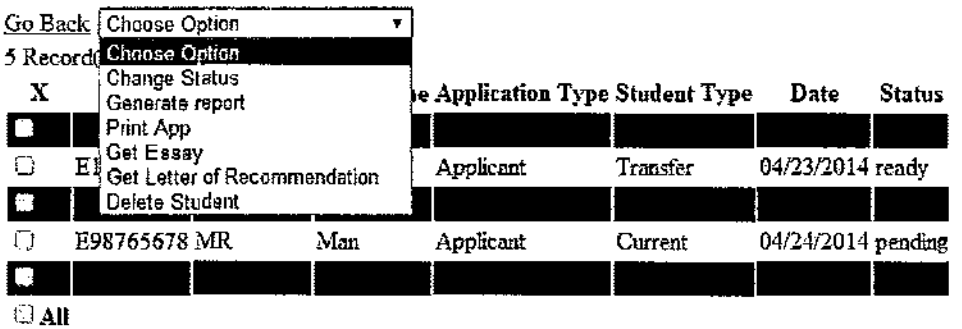
Eastern Michigan University Honors College
Reference Database Manager



- This part of the site is for managing the recommender's login database.
- The drop box is for selecting the recommender for a certain student.
- The Username text box is for creating or updating a recommender's username.
- The Password text box is for creating or updating a recommender's password.
- The Reference's Students EID is for linking the recommender to the student.
- The Reference's Number is for which recommendation they are. Some student's need two letters of recommendation which means they will need someone to be reference number one, and someone to be reference number two.
- The drop down menu at the bottom is for choosing which option you would like to do.
- Once finished, click the button at the bottom of the page to save the changes.

Admission Page (adminSearch.php)

Eastern Michigan University Honors College Admission Page



- This page is for manipulating the currently found students from the selected query.
- To select records, click the checkbox next to the student's EID. To select all the students found, click the checkbox next to the word All.
- The drop down menu allows you to choose the option you would like to use on the selected students.
- The drop down menu item, Change Status, is for changing the status of the selected students. You must also select from the next drop down menu, what the new status will be (Accepted – For students that you want to accept to the Honors College. Pending – For students who are still waiting for letters of recommendations. Ready – For students who have all the requirements ready to be reviewed. Not Accepted – For students who are not accepted to the Honors College).
- The drop down menu item, Generate Report, is for creating a spreadsheet of all the data known about the currently selected students.
- The drop down menu item, Print App, is for creating a pdf of a single selected

student. This feature will write the student's record to the Honors College application pdf. It will also attach the letters of recommendations and the student's essay to this pdf.

- The drop down menu item, Get Essay, is for displaying the currently selected student's essay.
- The drop down menu item, Get Letter of Recommendation, is for displaying the letters of recommendations and scores for the currently selected student. This page also has a feature where you can click a checkbox if you received the letter in a way other than through the site.
- The drop down menu item, Delete Student, is for deleting the currently selected students. After the user clicks Submit, a new page will pop up. This page will display the students that will be deleted. The administrator must enter their password twice to make sure that this was not an accident.